

Dynamisk analyse av sylindrisk olje demper system

Marius Øgård

Master i produktutvikling og produksjon

Innlevert: juni 2013

Hovedveileder: Torbjørn Kristian Nielsen, EPT

Medveileder: Dan Østling, Teeness

Norges teknisk-naturvitenskapelige universitet
Institutt for energi- og prosesseteknikk

EPT-M-2013-119

MASTEROPPGAVE

for

Stud.techn. Marius Øgård

Våren 2013

Strømning i sylindrisk oljedemper*Flow in cylindrical oil damper.***Bakgrunn og målsetting**

Sandvik Teeness utvikler og produserer vibrasjonsdempede verktøy for metallbearbeiding.

Det patenterte dempesystemet er passivt og baserer seg på å absorbere den laveste svingemodien i verktøyet vha et klassisk fjær-masse-dempe-system. Selve dempingen skjer ved at viskøs olje strømmes frem og tilbake i en sylindrisk kanal og det er denne oljestrømmen prosjektoppgaven dreier seg om.

Det foreligger en analytisk modell for dempingen i en slik struktur som stemmer bra for det normale arbeidsområdet for slike dempere.

Masteroppgaven tar utgangspunkt i prosjektarbeid utført høsten 2012. Det arbeidet omhandlet en kvasi-stasjonær løsning av et olje demper system.

Det teoretiske grunnlaget er begrenset av små bevegelser og denne oppgaven vil se på hvordan et slikt system vil oppføre seg under større bevegelser.

Opgaven tar for seg en dynamisk analyse av et olje demper system.

Opgaven bearbeides ut fra følgende punkter:

- 1 Litteraturstudie av problemstillingen, søk etter artikler som omhandler analytiske metoder for å løse strømningstilfellet.
- 2 Bygge videre på prosjektoppgavens resultater og se hvilke muligheter mer avanserte metoder som f.eks dynamisk meshing kan gi.
- 3 Noen av resultatene skal verifiseres gjennom praktiske forsøk. Her planlegges det å bruke vibrasjonstestestyr ved Teeness.

Senest 14 dager etter utlevering av oppgaven skal kandidaten levere/sende instituttet en detaljert fremdrift- og eventuelt forsøksplan for oppgaven til evaluering og eventuelt diskusjon med faglig ansvarlig/veileder. Detaljer ved eventuell utførelse av dataprogrammer skal avtales nærmere i samråd med faglig ansvarlig.

Besvarelsen redigeres mest mulig som en forskningsrapport med et sammendrag både på norsk og engelsk, konklusjon, litteraturliste, innholdsfortegnelse etc. Ved utarbeidelsen av teksten skal kandidaten legge vekt på å gjøre teksten oversiktlig og velskrevet. Med henblikk på lesning av besvarelsen er det viktig at de nødvendige henvisninger for korresponderende steder i tekst, tabeller og figurer anføres på begge steder. Ved bedømmelsen legges det stor vekt på at resultatene er grundig bearbeidet, at de oppstilles tabellarisk og/eller grafisk på en oversiktlig måte, og at de er diskutert utførlig.

Alle benyttede kilder, også muntlige opplysninger, skal oppgis på fullstendig måte. For tidsskrifter og bøker oppgis forfatter, tittel, årgang, sidetall og eventuelt figurnummer.

Det forutsettes at kandidaten tar initiativ til og holder nødvendig kontakt med faglærer og veileder(e). Kandidaten skal rette seg etter de reglementer og retningslinjer som gjelder ved alle (andre) fagmiljøer som kandidaten har kontakt med gjennom sin utførelse av oppgaven, samt etter eventuelle pålegg fra Institutt for energi- og prosesssteknikk.


Risikovurdering av kandidatens arbeid skal gjennomføres i henhold til instituttets prosedyrer. Risikovurderingen skal dokumenteres og inngå som del av besvarelsen. Hendelser relatert til kandidatens arbeid med uheldig innvirkning på helse, miljø eller sikkerhet, skal dokumenteres og inngå som en del av besvarelsen. Hvis dokumentasjonen på risikovurderingen utgjør veldig mange sider, leveres den fulle versjonen elektronisk til veileder og et utdrag inkluderes i besvarelsen.

I henhold til "Utfyllende regler til studieforskriften for teknologistudiet/sivilingeniørstudiet" ved NTNU § 20, forbeholder instituttet seg retten til å benytte alle resultater og data til undervisnings- og forskningsformål, samt til fremtidige publikasjoner.

Besvarelsen leveres digitalt i DAIM. Et faglig sammendrag med oppgavens tittel, kandidatens navn, veileders navn, årstall, instituttnavn, og NTNUs logo og navn, leveres til instituttet som en separat pdf-fil. Etter avtale leveres besvarelse og evt. annet materiale til veileder i digitalt format.

- Arbeid i laboratorium (vannkraftlaboratoriet, strømningssteknisk, varmeteknisk)
 Feltarbeid

NTNU, Institutt for energi- og prosesssteknikk, 14. januar 2013


Olav Bolland
Instituttleder


Torbjørn K. Nielsen
Faglig ansvarlig/veileder

Medveileder: Dan Østling, Teenes

Preface

This master thesis has been written at NTNU during the spring semester of 2013. The master thesis did not follow the expected plan due to the overwhelming job of analysing and verifying the simulation work so the plan has been altered in accordance with the supervisor.

All simulation work is done in OpenFOAM and some part of this thesis requires specific knowledge of this software.

It has been a great learning experience to write this thesis and I would first of all like to thank my supervisors Dan Østling & Torbjørn K. Nielsen for all the help. I would also like to thank Reidar Kristoffersen for help with specific flow-related questions.

I would also like to thank the 2013-team of Revolve NTNU for giving me the extra motivation during my last year, especially to Bjørn Vee and Magnus Krane for helping me with computer set-up and scripting respectively.

Last but not least I would like to thank my girlfriend for helping me with proof reading and for supporting me the entire time. Thank you Emeli.

My parents deserve a mention as well as they have always supported me and I am in forever gratitude to the help I have recieved from them.

The work done in this thesis is done in OpenFOAM 2.11, and a newer version is now available. This means that some of the code work done in this thesis needs to be changed in order to work on this new version.

Trondheim, June 7, 2013

Marius Øgård

Summary

This master thesis contains a fluid flow analysis of the dynamic system concerning two initial concentric cylinders where the cylinders can move in relation to each other. The thesis has been to develop a proper dynamic mesh model that involves dynamic motion of the system. This master thesis continues the work done in the project thesis *Flow in cylindrical oil damper* where the same problem was solved by the use of a quasi-static method.

A dynamic mesh model has been developed and it is based on an acceleration input that is translated into a dynamic mesh motion solver. The motion model takes in variables of amplitude, phase angle and frequency and uses a sinusoidal acceleration input. Other parameters that are used in the computational fluid model are viscosity and the geometry.

The dynamic model has been tested against a test-case. The test-case is an oscillating hydraulic piston case and no noticeable discrepancies were found.

The mathematical foundation for this case has been expanded further in this thesis and combined with the project thesis covers most of the known theory on this case.

The model was tested with different viscosities, geometries and frequencies, in order to find a non-dimensional number that can be used to scale this system for industrial usage.

Different simulation cases for geometry, frequency, amplitude and viscosity has been performed and most of the variables have been cross-referenced to understand how they affect the system.

The effect of the different variables on the system has been identified and summarized for the different cases. The trends of these results can be used for different geometries as long as the dimensionless numbers are kept within the same area.

All results from the simulations have been tested against the available theory and there are no inconsistencies for the results published in this thesis.

Norwegian summary

Denne masteroppgaven omhandler en dynamisk fluidanalyse av et system bestående av to i utgangspunktet konsentriske sylindre som kan bevege seg i forhold til hverandre. Oppgaven har tatt for seg utviklingen av en dynamisk mesh modell som brukes som grunnlag for den dynamiske bevegelsen. Masteroppgaven bygger videre på arbeidet som ble gjort i prosjektoppgaven *Flow in cylindrical oil damper* der det samme problemet ble løst med en kvasi-statisk metode.

Modellen som har blitt utviklet baserer seg på en akselerasjonsrespons som er transformert slik at den kan brukes på en dynamisk mesh løser. Modellen tar inn variabler som amplitude, fasevinkel og frekvens og den er basert på en sinusbasert akselerasjonsrespons.

Den dynamiske modellen har blitt testet mot et eksempeltilfelle. Testtilfellet var et hydraulisk stempel og ingen synlige avvik ble oppdaget under testing med den modellen.

Det matematiske grunnlaget for dette systemet har blitt utvidet enda mer i masteroppgaven og kombinert med det som ble definert i prosjektoppgaven dekker det mesteparten av den kjente teorien om dette systemet.

Videre så har den dynamiske modellen blitt testet med forskjellige viskositeter, geometrier og frekvenser i den hensikt å kartlegge et dimensjonsløst tall som kunne bli brukt til å skalere systemet.

Flere forskjellige simuleringstilfeller som geometri, frekvens, amplitude og viskositet har blitt undersøkt og mesteparten av disse variablene har blitt krysssjekket for å kartlegge hvordan de påvirker systemet.

De forskjellige variablene for systemet har blitt kartlagt og oppsummert for de undersøkte tilfellene som har vært analysert i denne oppgaven. Trendlinjene fra disse resultatene kan brukes for forskjellige geometrier, så lenge de dimensjonsløse tallene er innenfor de samme områdene som i denne undersøkelsen.

Alle resultatene fra simuleringene har blitt testet opp mot det som finnes av aktuell teori og det er ikke funnet noen ting som antyder at denne modellen ikke fungerer for dette formålet.

List of Figures

1.1	Picture of case geometry.	3
1.2	Reynolds number for all simulation cases performed in this master thesis.	5
1.3	Kinetic Reynolds number for all simulation cases performed in this thesis.	6
1.4	Polyhedral control volume (cell), From [17]	13
1.5	Mesh deformation,problem description, From [17]	14
2.1	Picture displaying the relation of how the moving inner cylinder is affecting the characteristic length.	19
3.1	Case geometry of the hydraulic cylinder comparison case	34
3.2	Mesh distribution of hydraulic cylinder case.	35
3.3	Velocity development at the beginning of the simulation for the hydraulic piston case.	36
3.4	Fully developed velocity profile for hydraulic cylinder case.	47
3.5	Picture from the simulated case of the hydraulic piston case.	48
3.6	Near-wall velocity overshoot. Taken from [7].	48
3.7	Velocity development as the cylinder is changing direction for the hydraulic piston case. B-value of 1.767	49
3.8	Overall velocity development for the hydraulic piston case.	50
3.9	Pressure development for case with varying alpha values.	53
3.10	Max pressure for case with varying alpha values.	54
3.11	Cylindrical moving mesh at initial state along y-axis.	55
3.12	Cylindrical moving mesh at compressed state along y-axis.	56
3.13	Cylindrical moving mesh at initial state along x-axis.	57
3.14	Cylindrical moving mesh at compressed state along x-axis.	57
3.15	Analytical solution for max pressure gradient as a function of angle.	59
3.16	Analytical solution for pressure gradient as a function of time.	60
3.17	Courant number during a dynamic mesh simulation.	61
3.18	Courant number for different frequencies at one viscosity value.	62
3.19	Courant number for different viscosities at one frequency value.	63

3.20	Chart of the B-value for all simulations done in this thesis.	64
3.21	Plot showing the velocity profile for case with B-value of 2.45	65
3.22	Plot showing the velocity profile for case with B-value of 0.15	66
3.23	Max velocity as a function of frequency and viscosity.	67
3.24	Courant number for simulations with different amplitudes.	68
3.25	Max Courant number plotted as a function of amplitude.	69
3.26	Max pressure force plotted against amplitude.	70
3.27	Max viscous force plotted against amplitude.	71
3.28	The relative force difference between pressure and viscous force plotted against amplitude.	72
3.29	Max simulated and calculated velocity component for different alpha values.	73
3.30	Example of force output for entire time interval for one case.	74
3.31	Example of force output for entire time interval for one case.	75
3.32	Relative maximum pressure force for variable alpha values.	76
3.33	Max pressure force for variable viscosity and frequency.	77
3.34	Relative maximum viscous force for variable alpha values.	78
3.35	Max viscous force for variable viscosity and frequency.	79
3.36	Relative force difference between pressure and viscous force for variable viscosity and alpha values.	80
3.37	Max relative pressure force for variable viscosity and frequency.	81
3.38	Max relative viscous force for variable viscosity and frequency.	82
3.39	Max velocity component for different viscosities and different alpha values.	83
3.40	Max velocity as a function of alpha. Plotted for different viscosities.	84
4.1	Courant number for adjustable time step simulation of the dynamic cylinder case.	86
4.2	Figure of dynamic cylinder case with symmetry plane along the y-axis.	87
B.1	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.5.	147
B.2	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.6.	148
B.3	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.7.	148
B.4	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.8.	149
B.5	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.85.	149
B.6	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.9.	150
B.7	Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.92.	150
B.8	Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.5.	151
B.9	Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.6.	151
B.10	Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.7.	152

B.11 Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.8.	152
B.12 Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.85.	153
B.13 Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.9.	153
B.14 Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.92.	154
B.15 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.5.	154
B.16 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.6.	155
B.17 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.7.	155
B.18 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.8.	156
B.19 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.85.	156
B.20 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.9.	157
B.21 Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.92.	157
B.22 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.5.	158
B.23 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.6.	158
B.24 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.7.	159
B.25 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.8.	159
B.26 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.85.	160
B.27 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.9.	160
B.28 Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.92.	161
B.29 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.5.	161
B.30 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.6.	162
B.31 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.7.	162
B.32 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.8.	163
B.33 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.85.	163
B.34 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.9.	164
B.35 Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.92.	164
B.36 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.5.	165
B.37 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.6.	165
B.38 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.7.	166
B.39 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.8.	166
B.40 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.85.	167
B.41 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.9.	167
B.42 Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.92.	168
B.43 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.5.	168
B.44 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.6.	169
B.45 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.7.	169

B.46 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.8.	170
B.47 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.85.	170
B.48 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.9.	171
B.49 Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.92.	171
B.50 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.5.	172
B.51 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.6.	172
B.52 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.7.	173
B.53 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.8.	173
B.54 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.85.	174
B.55 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.9.	174
B.56 Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.92.	175
B.57 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.5.	175
B.58 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.6.	176
B.59 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.7.	176
B.60 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.8.	177
B.61 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.85.	177
B.62 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.9.	178
B.63 Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.92.	178
B.64 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.5.	179
B.65 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.6.	179
B.66 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.7.	180
B.67 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.8.	180
B.68 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.85.	181
B.69 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.9.	181
B.70 Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.92.	182

List of Tables

2.1	Table with all relevant moving mesh boundary conditions for transversal motion of this case in OpenFOAM.	24
3.1	Table with all currently available dynamic mesh solvers for OpenFOAM.	51

Table of Contents

Preface	i
Summary	ii
Norwegian Summary	iii
Definitions & abbreviations	xii
1 Overview	1
1.1 Introduction	2
1.2 Model limitations	4
1.3 Non-dimensional study	4
1.3.1 Reynolds number	4
1.3.2 Kinetic Reynolds number	6
1.3.3 Strouhal number	7
1.4 Dynamic mesh	7
1.4.1 Spring analogy	7
1.4.1.1 Vertex method	8
1.4.1.2 Segment method	9
1.5 Moving mesh methods	10
1.5.1 Topological changes method	10
1.5.1.1 Primitive mesh modifiers	11
1.5.1.2 Topology modifiers	11
1.5.1.3 Dynamic mesh objects	12
1.5.2 Finite Volume Method	12
1.6 Mesh deformation validity	13
1.6.1 Mesh deformation criteria	14
1.7 Diffusivity models	14
2 Theory	17
2.1 Oscillating flows	18
2.2 Settling time	20
2.3 Stability criterias	21
2.4 Boundary conditions	22
2.5 Boundary motion	24

2.6	Modification of library	26
2.7	Mathematical Model	29
2.7.1	Pressure force contribution	29
2.7.2	Viscous force contribution	30
3	Model simulations	33
3.1	Verification of the simulation model	34
3.1.1	Case geometry	34
3.1.2	Velocity development	35
3.1.3	Mathematical foundation	36
3.1.4	Tested case	47
3.2	Simulations	50
3.2.1	Challenges and problems	52
3.2.2	Geometry simulation	53
3.2.3	Mesh	55
3.2.4	Velocity and pressure gradient distribution	58
3.2.5	Stability of simulations	60
3.2.6	Richardson's annular effect	64
3.2.7	Frequency simulation	66
3.2.8	Amplitude simulations	68
3.2.9	Velocity simulations	72
3.3	Force output	73
3.3.1	Pressure force	76
3.3.2	Viscous force	77
3.3.3	Relative force	79
3.4	Viscosity analysis	82
4	Conclusion	85
4.1	Optimizing of simulation case	86
4.1.1	Adjustable time step	86
4.1.2	Symmetry plane	87
4.1.3	Cell layer addition/removal	88
4.2	Practical experiment	89
4.3	Conclusion	90
4.3.1	Non dimensional study	90
4.3.2	Gradient model	90
4.3.3	Verification of model for the dynamic system	90
4.3.4	Further work	90
4.3.5	Notes	91
	Bibliography	92

A	Case files and Matlab scripts	95
A.1	U file for hydraulic piston case.	95
A.2	p file for hydraulic piston case.	97
A.3	pointMotionU file for hydraulic piston case.	99
A.4	blockMeshDict file for hydraulic piston case.	101
A.5	U file for dynamic simulation of the concentric cylinder case.	104
A.6	p file for dynamic simulation of the concentric cylinder case.	106
A.7	pointDisplacement file for dynamic simulation of the concentric cylinder case.	108
A.8	blockMeshDict file for dynamic simulation of the concentric cylinder case.	110
A.9	controlDict file for dynamic simulation of the concentric cylinder case.	114
A.10	forces file for dynamic simulation of the concentric cylinder case.	116
A.11	angularOscillatingDisplacement PointPatchVectorField.C file.	118
A.12	oscillatingDisplacementPointPatch VectorField.C file.	123
A.13	libMyFunctionDisplacementPointPatch VectorField.C file.	127
A.14	pointDisplacement file for use with modified library.	131
A.15	Matlab script for generating .dat file for use in pointMotionU file.	133
A.16	Matlab script for plotting of analytical solution for pressure gradient.	135
A.17	Matlab script for autogeneration of simulated velocity profiles.	138
A.18	Matlab script for autogeneration of simulated force plots.	142
B	Intermediate velocity plots from case with different viscosity and alpha value.	146

Definitions & abbreviations

Re Reynolds number

PDE Partial Differential Equation

CFD Computational Fluid Dynamics

PISO Pressure implicit with splitting of operators

SIMPLE Semi-implicit method for pressure-linked equations

OpenFOAM Open Field Operation and Manipulation

CFD Computational Fluid Dynamics

CV Control Volume

SFD Squeeze Film Damping

CDS Central Discretization Scheme

UDS Upwind Discretization Scheme

ALE-FEM Arbitrary Lagrangian Eulerian Finite Element Method

FVM Finite Volume Method

SCL Space Conservation Law

Nomenclature

α - Ratio of inner to outer diameter of cylinders $[\frac{D_1}{D_2}]$

γ - Ratio of outer radius to inner radius $[\frac{r_2}{r_1}]$

ρ - Density $[\frac{kg}{m^3}]$

V - Volume $[m^3]$

\mathbf{n} - Outward pointing unit normal vector

\mathbf{u} - Fluid velocity $[\frac{m}{s}]$

\mathbf{u}_s - Fluid velocity of the boundary surface $[\frac{m}{s}]$

γ_ϕ - Diffusion coefficient
 s_ϕ - Volume source/sink of ϕ
 S - Boundary surface [m^2]
 ϕ - General tensorial property
 P - Cell centroid
 f - Face of cell
 S_f - Area of face f [m^2]
 \mathbf{n}_f - unit normal of face
 N - Computational point(s)
 Δt - Time step [s]
 \dot{m} - Fluid mass flux [$\frac{m^3}{s}$]
 \dot{V} - Volumetric face flux [$\frac{m^3}{s}$]
 K_α - Constant used in derivation of flow profile [m]
 K_β - Constant used in derivation of flow profile [-]
 u - Velocity [m/s]
 U - Velocity [m/s]
 u^* - Non dimensional velocity [-]
 D_i - Inner cylinder diameter [m]
 D_o - Outer cylinder diameter [m]
 ω - Angular frequency [1/s]
 L_c - Characteristic length [m]
 t - time [s]
 t^* - time [s]

D_H - Hydraulic diameter [m]

ν - kinematic viscosity [$\frac{m^2}{s}$]

A - Area [m^2]

P_W - Wetted Perimeter [m]

U - Average velocity[m/s]

Chapter 1

Overview

1.1 Introduction

This thesis will continue on the work done in my project thesis *Flow in cylindrical oil damper*[14] in the autumn of 2012. The project thesis looked at a quasi-stationary solution for transversal movement of the inner cylinder in contrast to the outer cylinder in a concentric cylinder arrangement. The master thesis will look at the dynamics of this case by exploring the use of dynamic mesh methods.

The theoretical analysis available is restricted to small movements of the inner cylinder, and this thesis will explore dynamic methods for understanding what happens during large movements of the inner cylinder.

The master thesis is a dynamic study of a cylindrical oil damper system and will progress through the following points:

- A literary study of the problem description. A view on analytical methods for this flow problem.
- Continue from the results achieved in the project thesis and assessment of the possibilities that more advanced methods like for instance dynamic meshing can provide.
- The results will be verified through practical experiments, if necessary.

The case is to simulate dynamic flow development for two concentric cylinders with fluid between the two cylinders.

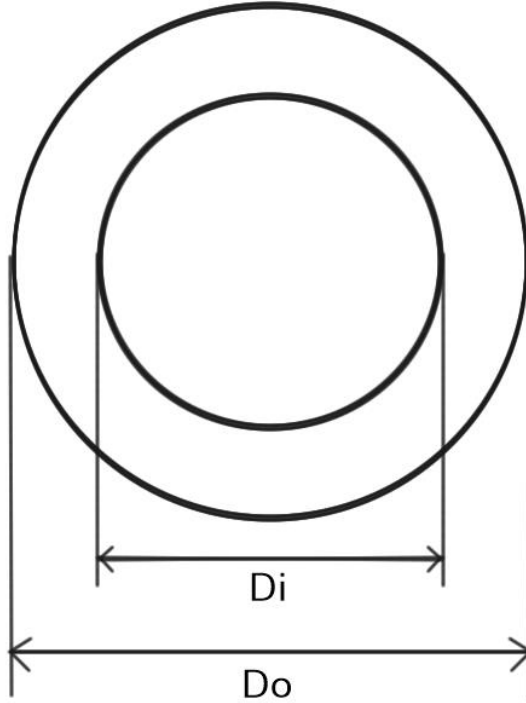


Figure 1.1: Picture of case geometry.

One of the main dimensionless parameters for the cylindrical oil damper system is the parameter α . This was used in the project thesis and will be used in the master thesis. It is defined in the following way:

$$\alpha = \frac{D_i}{D_o} \quad (1.1)$$

The work done in OpenFOAM that has been covered in the project thesis will not be elaborated on any further.[14]. Only new parts of this program will be covered. It is assumed that in order to set-up this dynamic case, knowledge of the use of OpenFOAM is necessary.

Certain processes such as the modification of the solver library is described in detail due to the fact that this is not available in the standard version of OpenFOAM and it is needed in order to simulate this case. All case files and the simulation set-up are available in the attachments.

1.2 Model limitations

Some limitations for the model have been set for the thesis. The main reason for this is due to the limited time span of the assignment as well as the complexity of dealing with laminar and turbulent models. The project thesis was limited to laminar flow and the same applies for this thesis. This will allow comparison of the data between the different models.

By limiting the model to the laminar area, the computational load of the simulations will be reduced compared to more advanced models. It will also limit the use of different solvers. The case itself is quite complex and the use of a dynamic mesh model with turbulent effects require more pre-study on that subject.

The frequencies and viscosities selected for this assignment has been selected based on knowledge of the oil-damper system and available fluids on the market.

The geometry values are also selected based on the values used in the project thesis and knowledge of the system in general.

1.3 Non-dimensional study

In order to understand and scale this case, a non-dimensional study should be performed. The most relevant non-dimensional numbers have been chosen based on the geometry and the possible effects that may arise during dynamic movements of this system.

1.3.1 Reynolds number

The Reynolds number is the most important non-dimensional number in fluid dynamics and gives an aspect of what type of flow character-

istics a given case defines. [7, p. 5] The Reynolds number is the ratio of inertial to viscous forces. The Reynolds number is defined as:

$$Re = \frac{U * L_C}{\nu} \quad (1.2)$$

For Reynolds numbers much lower than 1 there is creeping flow and for high Reynolds numbers there is turbulent flow. [7, p. 88]

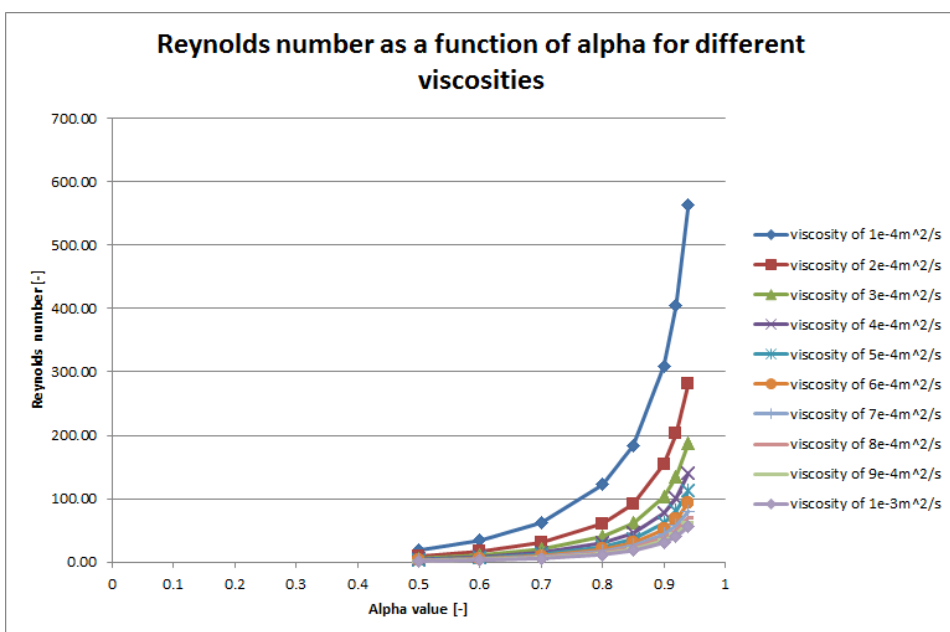


Figure 1.2: Reynolds number for all simulation cases performed in this master thesis.

All Reynolds numbers checked in this thesis are in the laminar area, the lowest Reynolds number is 1.91, so there are no cases of creeping flow.

It can be seen in figure 1.2 that the Reynolds number increases as the alpha value increases. Still, the cases are far from the turbulent transition area. A further study on this case can involve the turbulent

effects of this system. The turbulent effects are beyond the scope of this thesis.

1.3.2 Kinetic Reynolds number

There is a non-dimensional number for viscous effects in oscillating flows that is called the kinetic Reynolds number. [7, p. 127]

$$kinetic\ Re = \frac{\omega * h^2}{\nu} \tag{1.3}$$

For oscillating flows the flow may become turbulent if the kinetic Reynolds number is above 2000. [7, p. 128]

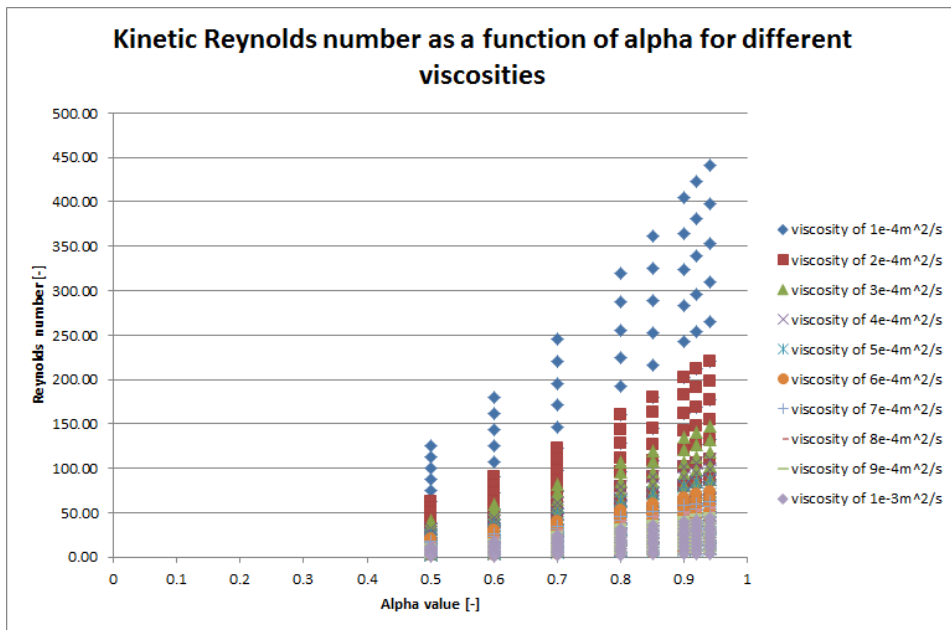


Figure 1.3: Kinetic Reynolds number for all simulation cases performed in this thesis.

It can be seen in figure 1.3 that there are many values for each viscosity at each alpha value. That is the different frequencies that are plotted

above each other for the different cases.

1.3.3 Strouhal number

The Strouhal number is based on the frequency of the vortex shedding behind a body. This vortex shedding can cause resonance if the shedding frequency is near the structural vibration frequency of the body. [6, p. 315]

$$St = \frac{\omega L_c}{U} \quad (1.4)$$

These three non-dimensional numbers can be used to indicate how the different parameters affect the fluid flow and where the stability areas are located for this case.

1.4 Dynamic mesh

In order to get a realistic simulation case for this case, a moving mesh method is needed. The cylindrical oil damper is used to remove vibrations in the tool holder, which causes a reduction of the vibrations in the tool itself. In order to induce forced vibrations on the system and inspect the transient effects of this, a moving mesh method is the only currently available method available in computational fluid dynamics (CFD).

1.4.1 Spring analogy

The mesh deformation that takes place during the time integration step of the computational fluid model must be defined in a valid way in order to keep the simulation stable. The spring analogy is a method used for mesh adaption and smoothing that defines a way of expanding and contracting the mesh. [16] Two different numerical methods for spring analogy are used; segment and vertex.

1.4.1.1 Vertex method

The vertex method defines the equilibrium length to be zero. The springs are considered linear and Hook's law defines the force at node i by the connected nodes j .

$$\vec{F}_i = \sum_{j=1} \alpha_{ij} (\vec{x}_j - \vec{x}_i) \quad (1.5)$$

In equation (1.5) the variable α_{ij} is the stiffness of the spring between node i and j . The sum notation is for all the connected nodes j . In order to satisfy equilibrium the force at every node i have to be zero. This will yield the iterative equation for \vec{x}_i^{k+1}

$$\vec{x}_i^{k+1} = \frac{\sum_{j=1} \alpha_{ij} \vec{x}_j^k}{\sum_{j=1} \alpha_{ij}} \quad (1.6)$$

Equation (1.6) will converge to static equilibrium for the system, but the mesh deformation does not need to converge between each time iteration. The spring model is a measure to keep the mesh from getting irregular during time iterations. This means that the number of iterations between each time iteration can be kept at a low number, and by that lowering the computational power required.

In the vertex spring method the stiffness value is set to a constant. The numerical value is set to unity for all i,j :

$$\alpha_{ij} = 1; \quad \forall i, j \quad (1.7)$$

The position of the internal mesh points is by that definition a mean of the connected nodes.

1.4.1.2 Segment method

In the segment method the equilibrium length is equal to the initial lengths of the segments. By applying Hook's law on the displacement on the nodes the force for each node will be as follows:

$$\vec{F}_i = \sum_{j=1} \alpha_{ij} (\vec{\delta}_j - \vec{\delta}_i) \quad (1.8)$$

The force on the node i will be zero for the static displacement as in the vertex method. This gives the following iterative equation:

$$\vec{\delta}_i^{k+1} = \frac{\sum_{j=1} \alpha_{ij} \vec{\delta}_j^k}{\sum_{j=1} \alpha_{ij}} \quad (1.9)$$

The boundary conditions for this equation is the known displacement of the boundaries. The spring stiffness can for instance be set to be proportional with the inverse of the segment length. Other models for the stiffness, such as quadratic or exponential, can be used.

A best practice is to perform an analysis of the spring stiffness before the segment method is chosen.

Since the spring stiffness is different from unity, there is a need to adjust the displacement after the iterations on the displacement.

$$\vec{x}_i^{new} = \vec{x}_i^{old} + \vec{\delta}_i^{k,final} \quad (1.10)$$

The need to store the displacement in the memory, makes the segment method more computationally demanding than the vertex method. Both spring models may have problems if the spring length is larger than a critical value, because then there is no stable equilibrium for the dynamic system. [21, p. 397]

1.5 Moving mesh methods

Moving mesh handling can be broken down into two different actions; *Mesh deformation* and *topological changes*. Mesh deformation is deformation of the mesh points through prescribed motion of the boundaries and not changing the structure in the mesh itself. Topological changes are where the connectivity of the mesh points, faces and cells are changed between the time-steps of the simulation.

There are two methods for discretising deforming meshes at the moment, and they are Arbitrary Lagrangian Eulerian Finite Element Method (ALE-FEM) and Finite Volume Method (FVM). These methods does not introduce any added discretisation error with deformable mesh operations compared to the use static mesh.

Topological changes in the mesh where points, cells and layers are added or removed between the time-steps require more computational power. This method introduces distribution and conservation errors associated with these changes, and the *deforming mesh* method is superior in a numerical and computational manner. [17]

For this case the deforming mesh method is suitable since the mesh does not require any topological changes during the simulation, and using the deformable mesh method the dynamic simulation will not introduce any discretisation error on this case.

1.5.1 Topological changes method

In cases where the mesh deformation method is not sufficient to satisfy the prescribed boundary conditions, the topological changes method needs to be used. Examples of cases where this is necessary are cases with rotating parts such as a mixer vessel case. For cases with rotating mesh connectivity, it is important that the mesh connection points are properly distributed between the rotor and the stator in order to reduce the computational error for this type of simulations. In OpenFOAM there are three components for handling of topological changes and they are: *primitive mesh operations*, *emph*topological modifiers and application-specific *dynamic mesh objects*.

1.5.1.1 Primitive mesh modifiers

Primitive mesh modifiers are the use of primitive operations such as addition, subtraction or modifications to modify a point, a face or a cell. More complex topological modifiers are built up on this language for mesh modification. This first level of functionalities gives access to basic operations that can be used on individual points, faces and cells. These operations are built into the mesh object and are independent of the discretisation.

This method is used in more complex algorithms in order to fully expand on the flexible choices of the primitive mesh modifiers. If executed manually it would be a very slow and tedious method where each operation has to be defined individually. This method is for instance used to completely rebuild meshes and then check for validity of the new mesh.

1.5.1.2 Topology modifiers

The primitive mesh modifiers are used in higher-level objects. These objects are called *mesh modifiers*. The mesh modifiers are used in triggering mechanisms that induce topological changes through primitive mesh operations. Layer addition/removal is one of these higher order object operations. The mesh will add or remove a layer when the layer has reached a maximum or minimum layer thickness respectively. The currently available types of topological modifiers for OpenFOAM are:

- Cell layer addition/removal - A layer will be added or removed to a mesh according to specified maximum and minimum layer thickness.
- Attach/Detach boundary - Internal faces will be converted to boundary patches. This operation can either be triggered automatically according to the solver or at pre-defined times specified by the user.
- Sliding interface - A pair of detached surfaces moving relative to each other. They will be attached when the two surfaces overlap.

The cell points will be disconnected during the point motion and connected again after the point motion.

There also exists error-driven adaptive mesh refinement methods [19], but they are not implemented into OpenFOAM at this time.

1.5.1.3 Dynamic mesh objects

The highest object level of topological changes is the *dynamic mesh objects*. Here you can specify multiple mesh modifiers for complex mesh motions. For instance, the mesh motion for a mixer vessel mesh can be prescribed only by the axis of rotation, the sliding interface and the rotational speed in rpm. All the complex mesh modifiers that need to be implemented for this to work in a topological way are automated by the use of this dynamic mesh object.

1.5.2 Finite Volume Method

The FVM with moving mesh is based on the integral form of the governing equation for an arbitrary moving volume V .

$$\frac{\partial}{\partial \mathbf{t}} \int_V \rho \phi \partial V + \oint_S \rho \mathbf{n} \cdot (\mathbf{u} - \mathbf{u}_s) \phi \partial S - \oint_S \rho \gamma_\phi \mathbf{n} \cdot \nabla \phi \partial S = \int_V S_\phi \partial V \quad (1.11)$$

The *space conservation law* (SCL) defines the relationship between the rate of change of the volume V and the boundary surface velocity of \mathbf{u}_s .

$$\frac{\partial}{\partial \mathbf{t}} \int_V \partial V - \oint_S \mathbf{n} \cdot \mathbf{u}_s \partial S = 0 \quad (1.12)$$

The polyhedral mesh space is discretised by splitting it into convex polyhedral bounded by convex polygons. The temporal property is split into time-steps and solved using normal time-stepping. A picture of an arbitrary polyhedral control volume is shown in figure 1.4 on the facing page

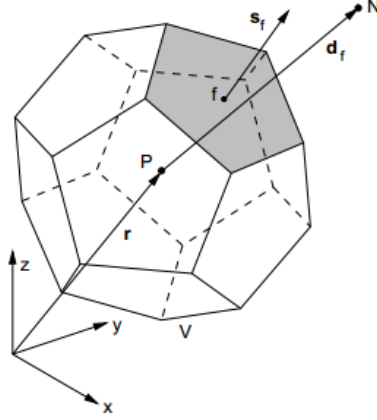


Figure 1.4: Polyhedral control volume (cell), From [17]

The second-order discretisation of equation (1.11) on the preceding page using a three time level scheme gives the following equation for cell P:

$$\frac{3\rho_P^{i+1}\phi_P^{i+1}V_P^{i+1} - 4\rho_P^i\phi_P^iV_P^i + \rho_P^{i-1}\phi_P^{i-1}V_P^{i-1}}{2\Delta t} + \sum_f \left(\dot{m}_f^{i+1} - \rho_f^{i+1} \dot{V}_f^{i+1} \right) \phi_f^{i+1} = \sum_f (\rho\gamma_\phi)_f^{i+1} S_f^{i+1} \mathbf{n}_f^{i+1} \cdot (\nabla\phi)_f^{i+1} + s_\phi^{i+1} V_P^{i+1} \quad (1.13)$$

The cell volumes and the volumetric face flux V_f is calculated from the geometric movements that is defined in the boundary conditions. This satisfies the discrete form of the Space Conservation Law. [18]

1.6 Mesh deformation validity

Figure 1.5 on the following page displays a general mesh deformation case. D is defined as the mesh domain and B as the bounding surface at a given time t. After a time step Δt the given domain has changed

into a new domain D' with a corresponding bounding surface B' . It is desired to link the domain D with D' , in order to minimize the distortion on the control volumes between the two domains.

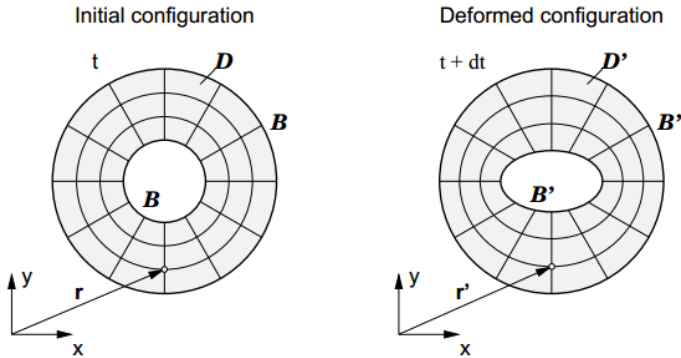


Figure 1.5: Mesh deformation, problem description, From [17]

The mesh validity constraint states that the domain is considered to be a linear elastic solid domain under deformation. The non-linear Piola-Kirchoff stress-strain formulation is often used and that is a computational expensive method to use. [17]

One common way of solving the mesh motion is with variational methods. [21] The Laplace equation with variable diffusivity is often used. The choice of diffusivity model need to be specified based on the case geometry.

1.6.1 Mesh deformation criteria

Mesh validity is a prerequisite for automatic mesh motion. This means that the mesh volumes need to be positive, the face areas need to be positive, face and cell convexness need to be preserved and within the mesh non-orthogonality bounds. [20]

1.7 Diffusivity models

The diffusivity models define how the mesh points are moved after solving the equations for each time step. The most common types of

diffusivity models are quality-based and distance-based models. The quality-based methods are models where the diffusion of the mesh is a function of the cell quantity. The following quality-based methods are:

- uniform - the mesh is moved uniformly during motion. It stretches and compresses uniformly.
- directional - the mesh manipulation is proportional to the direction of the motion. Two scalar coefficients needs to be defined for this method, they are respectively mean cell non-orthogonality and mean cell skewness.
- inverseDistance - a boundary is defined and the diffusivity of the mesh is based on the inverse of the distance from that boundary. This method is based on the inverse from the cell centre distance to the nearest boundary.
- inverseFaceDistance - Same as inverseDistance, but with the inverse from the cell face to the nearest boundary.
- inversePointDistance - Similar to the other inverse-methods, but this method is based on the inverse of the cell point to the nearest boundary.
- inverseVolume - Similar to the other inverse-methods, but in this method the method is based on the inverse of the cell volume.
- motionDirectional - one boundary is defined as the moving body and the mesh cells will be adjusted based on the motion of the moving body. Mean cell non-orthogonality and mean cell skewness also needs to be defined for this method.

The distance-based methods are used in conjunction with the inverse-quality-based methods:

- linear - this is the default option for distance-based methods and is selected by default. The mesh will move linearly inverse from the distance of the selected boundary.
- quadratic - The mesh diffusivity will be a quadratic function of the inverse of the selected boundary.

- exponential - Same principle as the two before mentioned methods, but with an exponential factor instead.

There is also a method called file where a file can be used to define the individual diffusivity of the cell points.

Chapter 2

Theory

2.1 Oscillating flows

The internal flow in the cylinder is oscillating and can be seen as a function of a sinusoidal function.

$$u = U * \sin(\omega t) \quad (2.1)$$

By non-dimensionalizing this velocity, it will look like:

$$\frac{u}{U} = u^* = \sin\left(\frac{\omega L_c}{U} t^*\right) \quad (2.2)$$

Where L_c is the characteristic length.

For the annular concentric cylinder case the characteristic length is given as the hydraulic diameter, D_H . The hydraulic diameter is a ratio of the wetted perimeter P_W and the cross-sectional area A .

$$D_H = \frac{4A}{P_W} \quad (2.3)$$

The hydraulic diameter is for a flow moving along the cross section of a annular pipe a ratio between the inner and outer diameter.

$$D_H = D_o - D_i \quad (2.4)$$

This is valid for flow moving along the cross section where the flow profile is equal in the angular direction. For the case with flow in the transverse direction this assumption might not be valid.

The part that appears along with the time variable in the sinus term is also known as the Strouhal number. Equation (1.4) on page 7 shows this term. This dimensionless number uses the shedding frequency of a body, and in this case this means the inner body. This vortex shedding

is something that must be considered and examined further once the simulation case is ready.

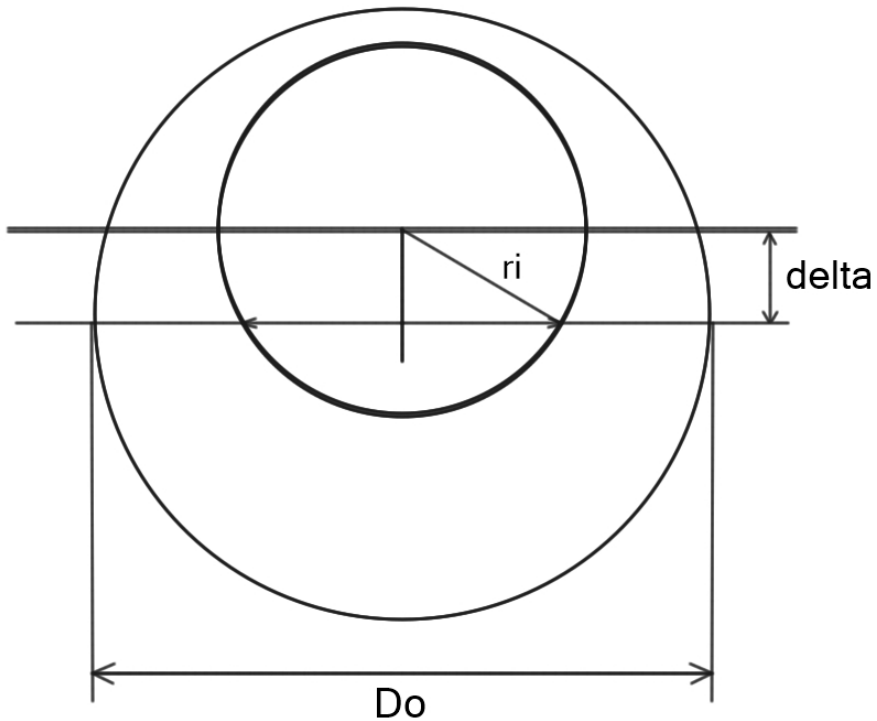


Figure 2.1: Picture displaying the relation of how the moving inner cylinder is affecting the characteristic length.

The characteristic length is in general a constant based on the geometry of the case. For annular flow in the transverse direction, the flow conditions changes continuously during the dynamic movement. Is the assumption of a constant characteristic length valid for this case? A dynamic characteristic length will be proposed in order to see how the Strouhal number with a dynamic characteristic length fits with known information about this system.

For annular pipe flow in the transverse direction, the hydraulic diameter can be given as a function depending on the location of the

inner cylinder.

The relation for this dynamic hydraulic diameter is given by the normal relation for the concentric annulus given by equation (2.4) on page 18 when the inner cylinder is in the neutral position. For a motion Δ away from the neutral position, I propose a hydraulic diameter of:

$$D_H = D_o - 2 * \sqrt{\frac{D_i^2}{2} - \Delta^2} \quad (2.5)$$

This assumption is made from the velocity gradient around the cross section for a moving inner cylinder. For this case the velocity is given by velocity found in the project thesis. [14] The same constants are used in this thesis:

$$u_\theta(r, \theta) = Ar + \frac{B}{r} + \frac{1}{\mu} \frac{\partial p}{\partial \theta} \frac{r \ln(r)}{2} \quad (2.6)$$

$$\frac{du_\theta(r, \theta)}{d\theta} = \frac{1}{2\mu} \frac{\partial^2 p}{\partial \theta^2} (r \ln(r)) \quad (2.7)$$

$$\frac{\partial p}{\partial \theta} = -\frac{\mu}{K_\alpha} \left(\frac{r_1/r_i}{\gamma - 1} + K_\beta \right) \quad (2.8)$$

Equation (2.7) for the velocity gradient can be found numerically and will be studied in order to see if this effect is something that can be used to understand the stability of the system.

2.2 Settling time

All fluids require time to fully develop their velocity profiles and in this case, with a dynamic system that is controlled by a given frequency, the need to understand this development is of importance.

In the project thesis it was clearly stated that this case was of laminar characteristics [14, p. 10] and for laminar pipe flow the entrance length is given by:

$$\frac{L_e}{d} = const \cdot Re_d \quad (2.9)$$

The constant has to be defined for a given geometry. By inserting the formula for the Reynolds number, it can be shown that the entrance length is only a function of velocity and the viscosity since the diameter for a pipe is the same as the hydraulic diameter of the pipe.

$$L_e = \frac{const \cdot U \nu d}{d_h} \quad (2.10)$$

$$L_e = const \cdot U \nu \quad (2.11)$$

The distance needed to fully develop the velocity profile increases with viscosity and velocity. Translated to this case it increases for higher viscosity and smaller gap distance given as the parameter of α .

In order to find the constant for this case relevant literature has to be considered and check if the entrance length for this geometry has been properly defined. For laminar flow in horizontal direction in a concentric annulus, various studies has been performed. The study by C. Nouar et al gives various formulas for the entrance lengths and also summarises different other studies on this subject. [15]

2.3 Stability criterias

The same stability criteria is valid for moving mesh simulation using *pimpleDyMFoam* as by using regular *pimpleFoam* and that is the Courant number since the dynamic mesh method does not add more numerical approximations to the solver. [20]

The Courant-Friedrichs-Lewy condition (CFL condition) is the stability criterion that needs to be fulfilled in order for the solution to converge. This criterion is from the upwind method and is the actual criterion that needs to be satisfied is equation (2.12). But for negligible diffusion the criterion can be simplified to equation (2.14).[1, p. 146]

$$\Delta t < \frac{1}{\frac{2\Gamma}{\rho(\Delta x)^2} + \frac{u}{\Delta x}} \quad (2.12)$$

Where the left term in the denominator is the time stability requirement for the Peclet number and the right term is the time stability requirement for the Courant number.

$$\Delta t < \frac{\rho(\Delta x)^2}{2\Gamma} \quad (2.13)$$

$$\Delta t < \frac{\Delta x}{u} \quad (2.14)$$

For the *pimpleDyMFoam* the stability criteria is a Courant number less than one as seen in equation (2.15).

$$C = \frac{u\Delta t}{\Delta x} \leq 1 \quad (2.15)$$

2.4 Boundary conditions

Selecting the proper boundary conditions is essential in order to get a realistic simulation and there are a couple of ways of doing this in OpenFOAM. The initial simulations were performed with a velocity input on the cylinder through an external file.

$$y(t) = A \sin(\omega * t) \quad (2.16)$$

$$v(t) = -\frac{A}{\omega} \cos(\omega t) \quad (2.17)$$

After further study on the subject it was clear that the velocity and displacement method for moving the inner cylinder was unphysical and the boundary condition needed in order to get a proper set-up is a sinusoidal acceleration on the system.

$$a(t) = A \sin(\omega t) \quad (2.18)$$

Since there is no way of inserting this condition into OpenFOAM directly the need to use either a velocity or displacement condition is necessary. By using equation (2.18) and adding the condition that $v(0) = 0$ the velocity function will become:

$$v(t) = \frac{A}{\omega} [1 - \cos(\omega t)] \quad (2.19)$$

The corresponding displacement function with the initial condition of $y(0) = 0$ is as follows:

$$y(t) = \frac{A}{\omega} \left[t - \frac{\sin(\omega t)}{\omega} \right] \quad (2.20)$$

Both the velocity and the displacement function are now scaled with the frequency as a parameter. This means that the simulations will have a relative amplitude based on the frequency.

It was decided to follow up on both possibilities for the boundary conditions in order to see how they performed. Since the mathematical functions are not the same as in the standard available conditions, the need to use external input files through the *timeVaryingFixedMappedValue* boundary type was initialized.

2.5 Boundary motion

In order to simulate a transversal moving inner cylinder in relative motion to the outer cylinder a dynamic movement has to be added on the boundary. There are a couple of ways to do this in OpenFOAM and they are listed in table 2.1.

Name	Description
pointDisplacement	The only types of valid boundary conditions for this file is: oscillatingDisplacement, angularOscillatingDisplacement, timeVaryingUniformFixedValue & timeVaryingMappedFixedValue
pointDisplacementU	The only types of valid boundary conditions for this file is: oscillatingVelocity, angularOscillatingVelocity, timeVaryingUniformFixedValue & timeVaryingMappedFixedValue

Table 2.1: Table with all relevant moving mesh boundary conditions for transversal motion of this case in OpenFOAM.

As shown in table 2.1 the boundary conditions are almost the same for both cases, the only difference is that one is based on the velocity of the surface and the other is based on the relative motion of the surface. *angularOscillatingDisplacement* takes in variables of amplitude, omega, axis of rotation, origin of movement, and angle0. The movement is based on the formula equation (2.21) and it can be seen in line 161 in attachment B.9.

$$angle = angle_0 + Amplitude \cdot \sin(\omega \cdot t) \quad (2.21)$$

oscillatingDisplacement only considers the amplitude and omega value. The direction of the movement is based on the vector of the amplitude. This boundary condition is a simplification of the *angularOscillatingDisplacement* and is chosen for initial simulations.

The *timeVarying* boundary conditions require a file for velocity inputs where the structure is given as time and velocity inputs. The structure is as follows:

```
(
(time1 (Ux Uy Uz))
(time2 (Ux Uy Uz))
(time3 (Ux Uy Uz))
)
```

This structure is based on the use of the vector solver *velocityLaplacian* in *dynamicMeshDict*. It is possible to use a scalar solver for the *velocityLaplacian* by adding the velocity vector at the end of this type, like *velocityLaplacianUx* for a scalar solver in the x-direction. The corresponding input file would be:

```
(
(time1 (Ux))
(time2 (Ux))
(time3 (Ux))
)
```

In order to create proper files for this use, a Matlab script was developed as shown in attachment B.10. This was used to simulate a inner moving cylinder and made for the vector solver *velocityLaplacian*. In order to keep this type of simulation stable the velocity inputs needed to be in the same scale as the Δt of the simulation or else the sudden velocity changes would be too big for the solver to handle. This way

of simulating the inner movement was used in the initial simulations, in order to verify the simulation settings.

The *timeVarying* boundary condition was discarded due to the fact that in order to simulate sinusoidal movements on the inner cylinder the velocity input needed to be a cosine input and the initial velocity step was a major problem for stability of the simulation case. This was the case even for quite small velocities. The simulation works, but the limitation in low amplitudes and low Δt values make this type of boundary condition not suited for this case.

2.6 Modification of library

In order to use the new function for the velocity and displacement there is a need to modify the boundary condition solver file. The use of external .dat files did not work properly and the solution is to modify the solver file for the boundary condition. In order to do this in a proper way, a new boundary library needs to be defined.

The details on how to do this will be explained quite thoroughly due to the fact that this is important to understand and know about. So all necessary commands to do this will be explained thoroughly here.

First the .C files from the library that is the basis for the modified library must be copied to another location, preferably the \$FOAM_RUN folder. The library that was chosen was the *oscillatingDisplacementPointPatchVectorField*.

```
cp -r $FOAM_SRC/fvMotionSolver/pointPatchFields/derived/  
oscillatingDisplacement $FOAM_RUN
```

In order to compile the new solver, the *Make* folder need to be copied to the new location.

```
cp -r $FOAM_SRC/fvMotionSolver/Make $FOAM_RUN/  
oscillatingDisplacement
```

The new solver folder need to be cleaned of old files by using the *wmake* command as shown in the next two steps.

```
cd $FOAM_RUN/oscillatingDisplacement
```

```
wclean
```

Then it is good practice to rename the files and folders so that the naming is not mixed with the original solver library. The solver name that was chosen was *MyFunctionDisplacement*. This is done in the following way.

```
cd $FOAM_RUN
```

```
mv oscillatingDisplacement libMyFunctionDisplacement
```

```
cd libMyFunctionDisplacement
```

```
mv oscillatingDisplacementPointPatchVectorField.C  
\libMyFunctionDisplacementPointPatchVectorField.C
```

```
mv oscillatingDisplacementPointPatchVectorField.H  
\libMyFunctionDisplacementPointPatchVectorField.H
```

The .C and .H files can now be modified to the new library by changing all the instances of *oscillatingDisplacement* to *libMyFunctionDisplacement*. This is done by the following commands. Also make sure that this is done in the *libMyFunctionDisplacement* folder.

```
sed -e 's/oscillatingDisplacement/libMyFunctionDisplacement/g'  
\libMyFunctionDisplacementPointPatchVectorField.C > tmp.C
```

```
mv tmp.C libMyFunctionDisplacementPointPatchVectorField.C
```

```
sed -e 's/oscillatingDisplacement/libMyFunctionDisplacement/g'  
\libMyFunctionDisplacementPointPatchVectorField.H > tmp.H
```

```
mv tmp.H libMyFunctionDisplacementPointPatchVectorField.H
```

In order to use the new library the file *files* in the *Make* folder need to be modified to look like this.

```
1 libMyFunctionDisplacementPointPatchVectorField.C
2
3 LIB=$(FOAM_USER_LIBBIN)/libMyFunctionDisplacement
```

The modification of `$FOAM_LIBBIN` to `$FOAM_USER_LIBBIN` is a preventive measure in order to secure that the original files stay the same during compilation.

The *options* file in the *Make* folder must include the following.

```
1 EXE_INC = \
2     -I$(LIB_SRC)/triSurface/lnInclude \
3     -I$(LIB_SRC)/meshTools/lnInclude \
4     -I$(LIB_SRC)/dynamicMesh/lnInclude \
5     -I$(LIB_SRC)/finiteVolume/lnInclude \
6     -I$(LIB_SRC)/fvMotionSolver/lnInclude
7
8 LIB_LIBS = \
9     -ltriSurface \
10    -lmeshTools \
11    -ldynamicMesh \
12    -lfiniteVolume
```

Before the library is compiled it is important to do the modification of the *libMyFunctionDisplacementPointPatchVectorField.C* file. The file is changed accordingly as in lines 111 to 113 in attachment B.11. The library is ready to compile and that is done in the following way.

```
cd $FOAM_RUN/libMyFunctionDisplacement
```

```
wmake libso
```

The compiling should now work without errors and the new library is now ready to use. In order to make the library available to the solver the following line has to be added to the *controlDict* file.

```
libs("libMyFunctionDisplacement.so");
```

The library can now be selected as a type in the boundary condition. This can be seen in attachment B.13 line 25.

2.7 Mathematical Model

The mathematical foundation for this case is available in the project thesis by Øgård. [14]

2.7.1 Pressure force contribution

By continuing the work started in the project thesis, and using the expression for the pressure gradient, an expression for the pressure can be found.

$$P(\phi, t) - P(-\frac{\pi}{2}) = \int_{-\frac{\pi}{2}}^{\phi} \frac{\partial p}{\partial \phi} d\phi = -\mu u_y(t) \int_{-\frac{\pi}{2}}^{\phi} \left(\frac{r_1}{r_2 - r_i(\phi, t)} + K_{\beta}(\phi, t) \right) \frac{\cos(\phi)}{K_{\alpha}(\phi, t)} d\phi \quad (2.22)$$

The pressure difference is of interest and for that reason the reference pressure $P(-\frac{\pi}{2})$ is set to zero for all values of t. The pressure distribution along the clearance can be used to find the active force counteracting the movement of the inner cylinder.

For symmetrical reasons the only component of the pressure that counteracts the inner cylinder movement is the y-component. The pressure force per length unit is then given by:

$$\frac{F_{p,y}(t)}{L} = -2 \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} P(\phi, t) r_1 \sin(\phi) d\phi = -\mu u_y(t) C_p(t) \quad (2.23)$$

Where the constant C_p is defined as:

$$C_p(t) = 2 \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \left[\int_{-\frac{\pi}{2}}^{\phi} \left(\frac{r_1}{r_i} + K_\beta \right) \frac{-r_1}{K_\alpha} \cos(\phi) d\phi \right] \sin(\phi) d\phi \quad (2.24)$$

This result need to be compared with the simulated values for different geometries. Equation (2.23) on the previous page may be solved numerically in order to get a time-dependent solution for a given case.

2.7.2 Viscous force contribution

In order to find the viscous force contribution on the inner cylinder, the expression for the viscous stress in cylindrical coordinates needs to be applied.

$$\tau_{r\phi} = -\mu r \frac{\partial}{\partial r} \left(\frac{u_\theta}{r} \right) \Big|_{r=r_i} = 2\mu \frac{B}{R_i^2} - \frac{1}{2} \frac{\partial p}{\partial \phi} \quad (2.25)$$

The same symmetry argument applies for the viscous force as well as as for the pressure force. The y-component of the viscous force per length unit is:

$$\frac{F_{s,y}(t)}{L} = -2 \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} r_i \tau(\phi, t) \cos(\phi) d\phi = -\mu u_y(t) C_s(t) \quad (2.26)$$

Where the constant C_s is defined in the following way:

$$C_s(t) = 2 \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \left[-\frac{r_1}{\alpha} \left(\frac{r_1}{r_i} + K_\beta \right) \right] \quad (2.27)$$

The structure of the viscous force is of similar structure as the pressure force and the only variation is in the variables C_s & C_p . The values for $K_\beta, K_\alpha, \gamma$ & r_i is defined in the following way:

$$r_i = \sqrt{r_1^2 + \Delta y^2(t) + 2r_1\Delta y(t)\sin(\theta)} \quad (2.28)$$

$$\gamma = \frac{r_2}{r_i} \quad (2.29)$$

$$K_\alpha = \frac{1}{4(r_2 - r_i)} \left[\frac{2r_i^2 r_2^2}{r_2^2 - r_i^2} \ln^2\left(\frac{r_2}{r_i}\right) - \frac{r_2^2 - r_i^2}{2} \right] = \frac{r_2}{4(\gamma - 1)} \left[\frac{2\gamma \ln^2(\gamma)}{\gamma^2 - 1} - \frac{\gamma^2 - 1}{2\gamma} \right] \quad (2.30)$$

$$K_\beta = \frac{r_i}{2(r_2 - r_i)} \left[\frac{2r_2^2}{r_2^2 - r_i^2} \ln\left(\frac{r_2}{r_i}\right) - 1 \right] = \frac{1}{2(\gamma - 1)} \left[\frac{2\gamma^2 \ln(\gamma)}{\gamma^2 - 1} - 1 \right] \quad (2.31)$$

The force expressions for the viscous and pressure force will be compared with the results from the force calculations performed during the simulations.

Chapter 3

Model simulations

3.1 Verification of the simulation model

In order to verify that the simulations performed in this thesis are correct, the need for a comparison case was necessary. The comparison case is a moving hydraulic cylinder. It contains all the necessary requirements to be used as a comparison case. A moving mesh, small gap size effects and known mathematical foundation. [7]

3.1.1 Case geometry

The case geometry was defined as shown in figure figure 3.1 and all the files that explains the boundary conditions and the mesh for this case are in attachment A.1 through A.4.

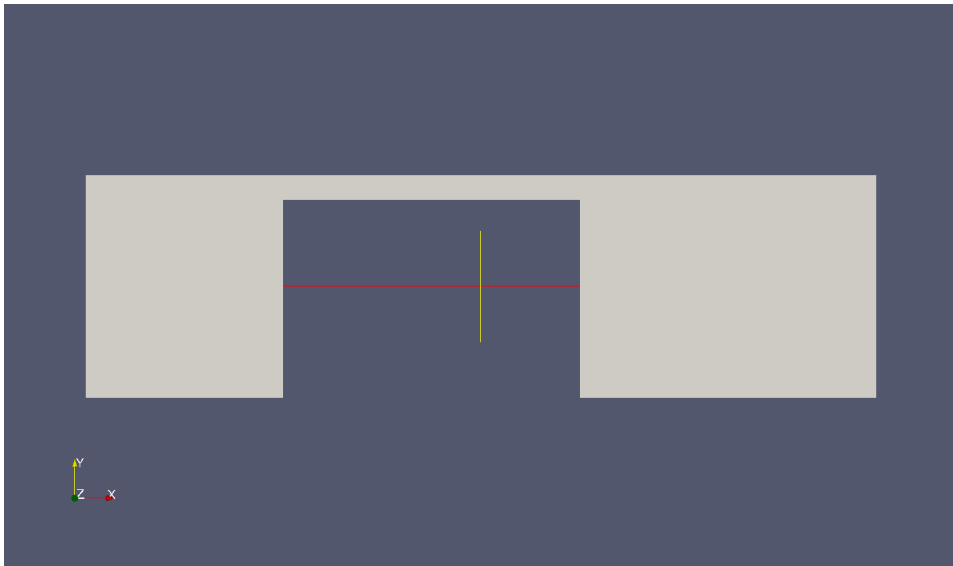


Figure 3.1: Case geometry of the hydraulic cylinder comparison case

The mesh was designed with increased focus on the flow in the small channel between the chambers on each side, since almost all the velocity changes are in this area. This can be seen in figure 3.2 on the next page.

The thick upper line in figure 3.2 on the facing page is part of the

mesh, but since it is so dense you can not properly see the grid. The detailed information of the mesh is given in attachment A.4

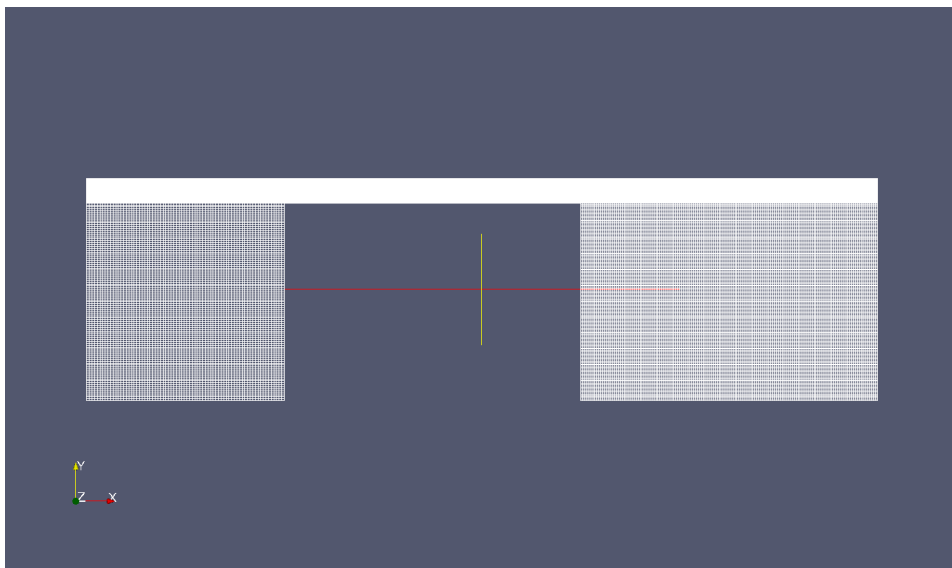


Figure 3.2: Mesh distribution of hydraulic cylinder case.

3.1.2 Velocity development

In order to validate the results from this case there is a need to compare the simulated velocity profiles with analytical results. First of all there is a need to validate that the developed velocity profiles look as they should for this case. The flow should develop to a parabola with a constant velocity in the other direction of the flow along the moving wall. The mass flow should also be a ratio of the channel width compared to the hydraulic cylinder times the moving velocity of the cylinder as shown in equation (3.1).

$$U_{channel,avg} = U_{piston} * \frac{h_{piston}}{h_{channel}} \quad (3.1)$$

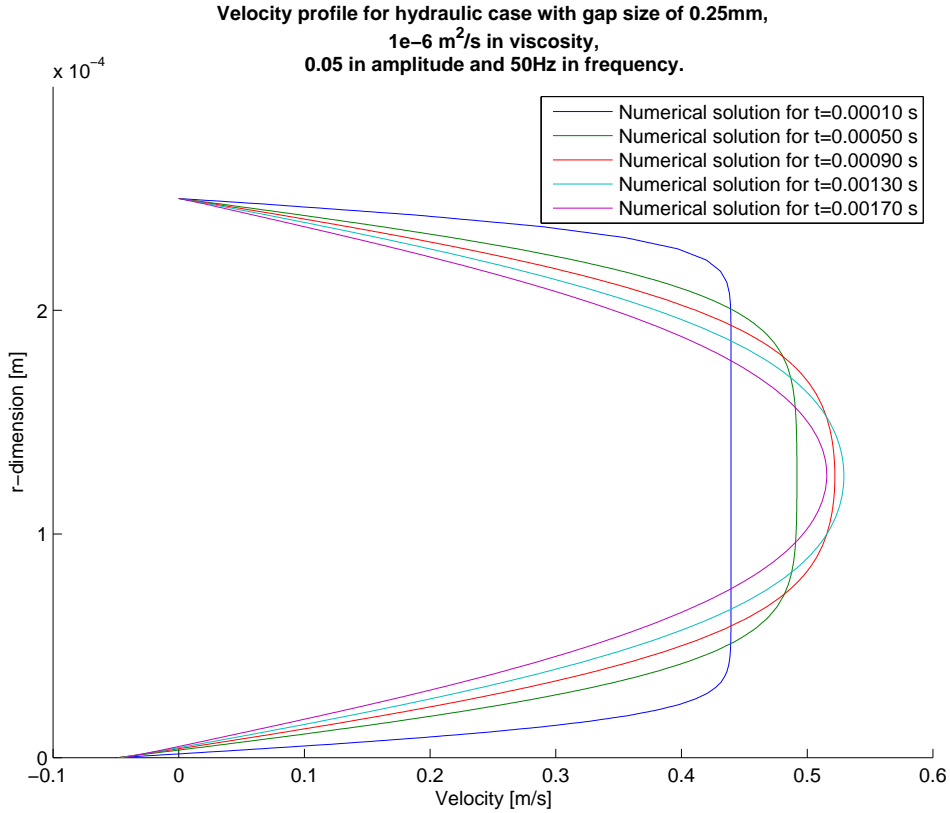


Figure 3.3: Velocity development at the beginning of the simulation for the hydraulic piston case.

The initial velocity profile for the simulation of the hydraulic piston case looks as in figure 3.3

3.1.3 Mathematical foundation

In order to compare the results from the simulated case, the velocity profile for the current case needs to be defined mathematically. The flow in the channel is a function of y and time for this case. It is assumed that there is no velocity in the y and z -direction.

$$u = u(y, t) \text{ and } v = w = 0 \quad (3.2)$$

The convective acceleration and the external forces are assumed to be zero for this case. The momentum equation then becomes:

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \frac{dp}{dx} + \nu \frac{\partial^2 u}{\partial y^2} \quad (3.3)$$

By using dimensionless variables as defined in equation (3.4)

$$\tilde{u} = \frac{u}{U} \quad \tilde{x} = \frac{x}{h} \quad \tilde{y} = \frac{y}{h} \quad \tilde{p} = \frac{p}{\rho U^2} \quad \tilde{t} = \frac{t\nu}{h^2} \quad (3.4)$$

The momentum equation become as in equation (3.5)

$$\frac{\partial \tilde{u}}{\partial \tilde{t}} = -\frac{d\tilde{p}}{d\tilde{x}} * Re + \frac{\partial^2 \tilde{u}}{\partial \tilde{y}^2} \quad (3.5)$$

For simplicity the tilde notation is changed with the regular notation.

$$\frac{\partial u}{\partial t} = -\frac{dp}{dx} * Re + \frac{\partial^2 u}{\partial y^2} \quad (3.6)$$

The total channel length is over ten times the length of the total channel height and an approximation of infinite channel length is used for this case. Entrance effects are also ignored for this case.

By setting $a = -\frac{dp}{dx} * Re$, equation (3.6) becomes a separable Partial Differential Equation (PDE).

$$\dot{u} = a + u'' \quad (3.7)$$

The oscillating pressure gradient is defined as:

$$\frac{dp}{dx} = -K \cos(\omega t) \quad (3.8)$$

By using the same dimensionless variables as in equation (3.4) on the preceding page, equation (3.8) becomes:

$$\frac{d\tilde{p}}{d\tilde{x}} = -K \cos(\tilde{\omega} t) \quad (3.9)$$

The boundary conditions and the initial condition for equation (3.7) on the preceding page is as follows:

$$\tilde{u}(y = -h, t) = u(-1, t) = 0 \quad (3.10)$$

$$\tilde{u}(y = h, t) = u(1, t) = 0 \quad (3.11)$$

$$\tilde{u}(y, \tilde{t} = 0) = u(y, 0) = 0 \quad (3.12)$$

First the solution for $u_1(y, t)$ of the homogeneous equation with the boundary and initial conditions as defined in equation (3.10) to equation (3.12) on the current page must be found.

$$\dot{u} = u'' \quad (3.13)$$

Equation (3.13) can be solved by *the method of separation of variables*. The solution is on the form:

$$u_1(y, t) = F(y) * G(t) \quad (3.14)$$

That can be rewritten as:

$$F * \dot{G} = F'' * G = \frac{\dot{G}}{G} = \frac{F''}{F} \quad (3.15)$$

The relation in equation (3.15) must be equal to a constant since they are dependent on different variables. This constant must be negative, since a positive constant gives the trivial solution of $u=0$.

$$\frac{\dot{G}}{G} = \frac{F''}{F} = -p^2 \quad (3.16)$$

The problem is now reduced to two ordinary linear differential equations.

$$F'' + p^2 * F = 0 \quad (3.17)$$

$$\dot{G} + p^2 * G = 0 \quad (3.18)$$

The general solution for equation (3.17) is on the form:

$$F = A \cos(py) + B \sin(py) \quad (3.19)$$

In equation (3.19) the constant $A \neq 0$ because the solution will become the unphysical solution of $u(0, t) = 0$.

From the boundary and initial conditions in equation (3.10) to equation (3.11) on the facing page, the solution for equation (3.17) will become zero. Combined with the fact that $A \neq 0$ the only solution is that $B = 0$. This gives the following equation.

$$F(\pm 1) = A \cos(\pm p) = 0 \quad (3.20)$$

Equation (3.20) will only be satisfied when the cosine term is zero and that is for $p = \frac{\pi}{2}, \frac{3\pi}{2}, \frac{5\pi}{2}, \dots$

A more compact way of writing the expression for p is as follows:

$$p = (2n - 1) \frac{\pi}{2}, \quad n = 1, 2, 3, 4, \dots \quad (3.21)$$

The p-value will be defined as p_n for equation (3.22). The solution for that equation will then be on the form:

$$G_n(t) = B_n e^{-p_n^2 t} \quad (3.22)$$

B_n act as a constant in equation (3.22). The total solution is the combined solution of equation (3.20) and equation (3.22):

$$u_1(y, t) = F_n(y) * G(t) = \sum_{n=1}^{\infty} B_n \cos(p_n y) * e^{-p_n^2 t}, \quad n = 1, 2, 3, \dots \quad (3.23)$$

Equation (3.23) need to satisfy the initial condition in equation (3.12) on page 38. This reduces u_1 to:

$$u_1(y, t = 0) = \sum_{n=1}^{\infty} B_n \cos(p_n y) = 0 \quad (3.24)$$

B_n can now be found by a cosine Fourier expansion for the initial velocity.

$$B_n = \int_{-1}^1 u(y, t = 0) * \cos(p_n y) dy \quad (3.25)$$

Since $u(y, t = 0)$ is zero, B_n will be zero for all values of n . Equation (3.23) on the preceding page is then reduced to $u_1 = 0$.

The solution for $u_2(y, t)$ for the inhomogenous equation (3.7) with boundary and initial conditions from equation (3.10) to equation (3.12) on page 38 will be on the form:

$$u_2(y, t) = \sum_{n=1}^{\infty} \phi_n(t) \cos(p_n y) \quad (3.26)$$

For the constant a in equation (3.7) the expansion will be on the form:

$$a = \sum_{n=1}^{\infty} \gamma_n(t) \cos(p_n y) \quad (3.27)$$

γ_n is defined as:

$$\gamma_n = \int_{-1}^1 a * \cos(p_n y) dy = \frac{-2a}{p_n} * (-1)^n \quad (3.28)$$

By inserting equation (3.26) to equation (3.27) on the current page into equation (3.7) an ordinary differential equation (ODE) for $\phi_n(t)$ is derived.

$$\frac{d\phi_n}{dt} + p_n^2 \phi_n = \gamma_n, \quad n = 1, 2, 3, \dots \quad (3.29)$$

In order for equation (3.29) on the preceding page to satisfy the initial condition. $\phi_n(t = 0)$ must be zero.

$$\phi_n(t = 0) = 0 \quad (3.30)$$

The *integrating factor* for equation (3.29) on the previous page is:

$$m = e^{\int p_n^2 dt} = e^{p_n^2 t} \quad (3.31)$$

The solution will be on the form:

$$m\phi_n = \int m\gamma_n dt \quad (3.32)$$

The pressure gradient is time dependent and entering the ODE through a in the expression for γ_n in equation (3.28). The solution will become:

$$m\phi_n = \int m\gamma_n dt = \frac{-2}{p_n}(-1)^n \int ae^{p_n^2 t} dt \quad (3.33)$$

By inserting equation (3.8) on page 38 into equation (3.33) the expression will be:

$$m\phi_n = \frac{-2}{p_n}(-1)^n \int ae^{p_n^2 t} dt = -\frac{2K}{p_n}(-1)^n \int \cos(\omega t)e^{p_n^2 t} dt \quad (3.34)$$

By solving the integral in equation (3.34) the solution will take the following form:

$$\begin{aligned}
m\phi_n &= -\frac{2K}{p_n}(-1)^n \int \cos(\omega t) e^{p_n^2 t} = \\
&-\frac{2K}{p_n}(-1)^n \frac{e^{p_n^2 t}}{p_n^4 + \omega^2} (p_n^2 * \cos(\omega t) + \omega * \sin(\omega t)) + C \quad (3.35)
\end{aligned}$$

By inserting the initial condition in equation (3.30) into equation (3.35) the constant C may be found.

$$C = -\frac{2K}{p_n}(-1)^n \frac{1}{p_n^4 + \omega^2} (p_n^2) \quad (3.36)$$

By inserting the constant C into equation (3.35), the expression for ϕ_n become:

$$\phi_n = -\frac{2K}{p_n}(-1)^n \frac{1}{p_n^4 + \omega^2} \left(p_n^2 * e^{-p_n^2 t} - p_n^2 * \cos(\omega t) - \omega * \sin(\omega t) \right) \quad (3.37)$$

Now the solution for $u(y, t)$ can be found.

$$u(y, t) = \sum_{n=1}^{\infty} -\frac{2K}{p_n}(-1)^n \frac{\cos(p_n y)}{p_n^4 + \omega^2} \left(p_n^2 * \cos(\omega t) + \omega * \sin(\omega t) - p_n^2 * e^{-p_n^2 t} \right) \quad (3.38)$$

By ignoring the start-up of the flow and only looking at the average velocity $U(t)$, the expression will reduce to:

$$U(t) = \frac{1}{2} \int_{-1}^1 u(y, t) = \sum_{n=1}^{\infty} \frac{2K}{p_n^2} \frac{p_n^2 \cos(\omega t) + \omega \sin(\omega t)}{p_n^4 + \omega^2} \quad (3.39)$$

When $\omega = 0$ the limit of equation (3.39) on the previous page is:

$$U(t) = \frac{1}{2} \int_{-1}^1 u(y, t) = \sum_{n=1}^{\infty} \frac{2K}{p_n^4} \quad (3.40)$$

The expression in equation (3.40) is similar to the quasi-static Poiseuille flow. For large values of ω the flow will have a time delay due to the pressure gradient and reach a maximum phase angle of 90° . [7, p. 128]

Another phenomenon are the near wall velocity overshoot of this type of flow, also known as the *Richardson's annular effect*. [7, p. 129]

There is another way of deriving this type of flow and it is based on the use of the Bessel function.

$$\frac{dp}{dx} = -\rho K e^{i\omega t} \quad (3.41)$$

In equation (3.41) the use of both the real and the imaginary part has been used for the pressure gradient.

$$e^{i\omega t} = \cos(\omega t) + i \sin(\omega t) \quad (3.42)$$

$$i = \sqrt{-1} \quad (3.43)$$

By considering the long-term steady oscillating flow, neglecting the start-up effects as well as the no-slip condition, the expression in equation (3.44) on the facing page will arise.

$$u = \frac{K}{i \omega} e^{i\omega t} \left[1 - \frac{J_0 \left(r \sqrt{\frac{-i\omega}{\nu}} \right)}{J_0 \left(r_0 \sqrt{\frac{-i\omega}{\nu}} \right)} \right] \quad (3.44)$$

Where J_0 is the Bessel function. This expression have the overlapping solution as shown in equation (3.45) & equation (3.46).

$$J_0(z) \approx 1 - \frac{z^2}{4} + \frac{z^4}{64} - \dots \quad z < 2 \quad (3.45)$$

$$J_0(z) \approx \sqrt{\frac{2}{\pi z}} \cos \left(z - \frac{\pi}{4} \right) \quad z > 2 \quad (3.46)$$

Dimensionless variables as shown in equation (3.47) are used for the rest of the derivation.

$$r^* = \frac{r}{r_0} \quad \omega^* = \frac{\omega r_0^2}{\nu} \quad u^* = \frac{u}{u_{max}} \quad (3.47)$$

Note that the dimensionless variable for the *kinetic Reynolds number* are one of the dimensionless variables beeing used.

u_{max} is defined in the following way:

$$u_{max} = \frac{K r_0^2}{4\nu} \quad (3.48)$$

By introducing the dimensionless variables as shown in equation (3.47) the solution can be rewritten too:

$$\frac{u}{u_{max}} \approx (1 - r^{*2} \cos(\omega t) + \frac{\omega^*}{16}(r^{*4} + 4r^{*2} - 5)\sin(\omega t) + O(\omega^{*2}) \quad \omega^* < 4 \quad (3.49)$$

$$\frac{u}{u_{max}} \approx \frac{4}{\omega^*} \left[\sin(\omega t) - \frac{e^{-B}}{\sqrt{r^*}} \sin(\omega t - B) \right] + O(\omega^{*-2}) \quad \omega^* > 4 \quad (3.50)$$

Where B is defined in the following way:

$$B = (1 - r^*) \sqrt{\frac{\omega^*}{2}} \quad (3.51)$$

By taking the average of equation (3.50) over one cycle, the mean square velocity can be obtained.

$$\frac{\overline{u^2}}{\frac{K^2}{2\omega^2}} = 1 - \frac{2}{\sqrt{r^*}} e^{-B} \cos(B) + \frac{e^{-2B}}{r^*} \quad (3.52)$$

Richardson's annular effect with the near wall mean velocity overshoot takes place when:

$$\cos(B) + \sin(B) \approx e^{-B}, \quad B \approx 2.284 \quad (3.53)$$

3.1.4 Tested case

The fully developed flow for a case with a piston to gap-ratio of 8 is shown in figure 3.4. The initial velocity of the system was $0.05 \frac{m}{s}$. The average mass flow measured for this system was 0.3993. The theoretical number for this case is 0.4. The small discrepancy is most likely due to bad sampling or minor numerical deviations.

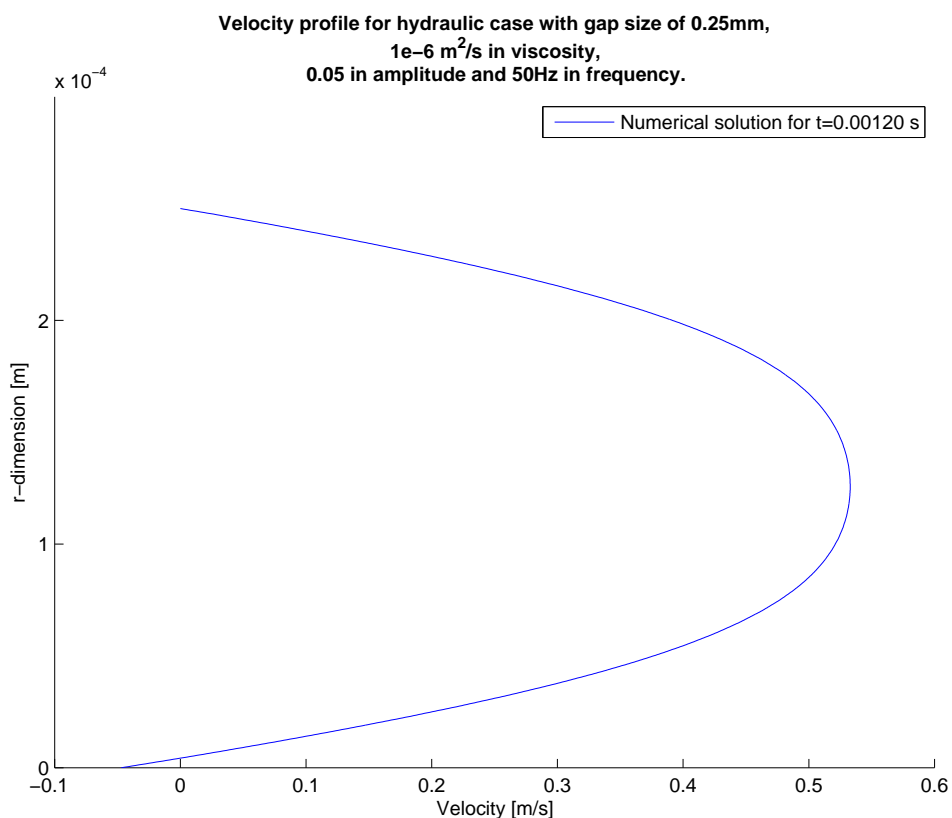


Figure 3.4: Fully developed velocity profile for hydraulic cylinder case.

A picture of the simulated case is shown in figure 3.5 on the next page and here it can be seen that this case has vortex shedding between the piston and the open chamber. An interesting fact is that no signs of vortex shedding has been found in the concentric annulus case.

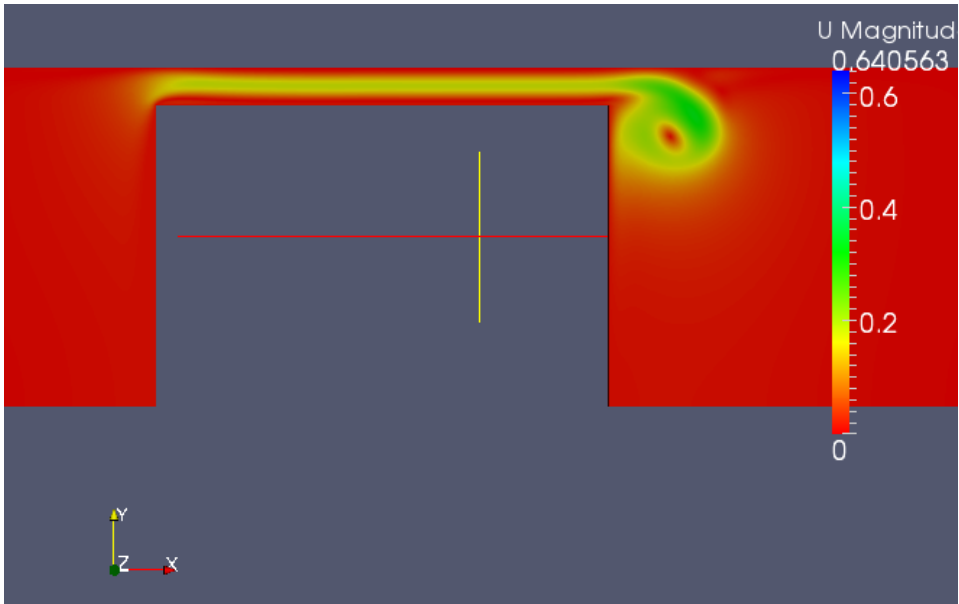


Figure 3.5: Picture from the simulated case of the hydraulic piston case.

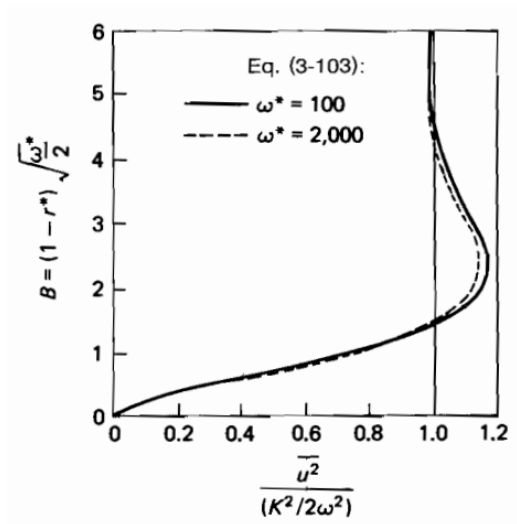


Figure 3.6: Near-wall velocity overshoot. Taken from [7].

The B-value for this hydraulic test case was calculated to be 1.767.

A plot of the equation for *Richardson's annular effect* is shown in figure 3.6 on the facing page and according to that figure this value should give a near-wall velocity overshoot.

It is evident in figure 3.7 that there is a near-wall velocity overshoot for this case and it is in accordance with the theory in this field.

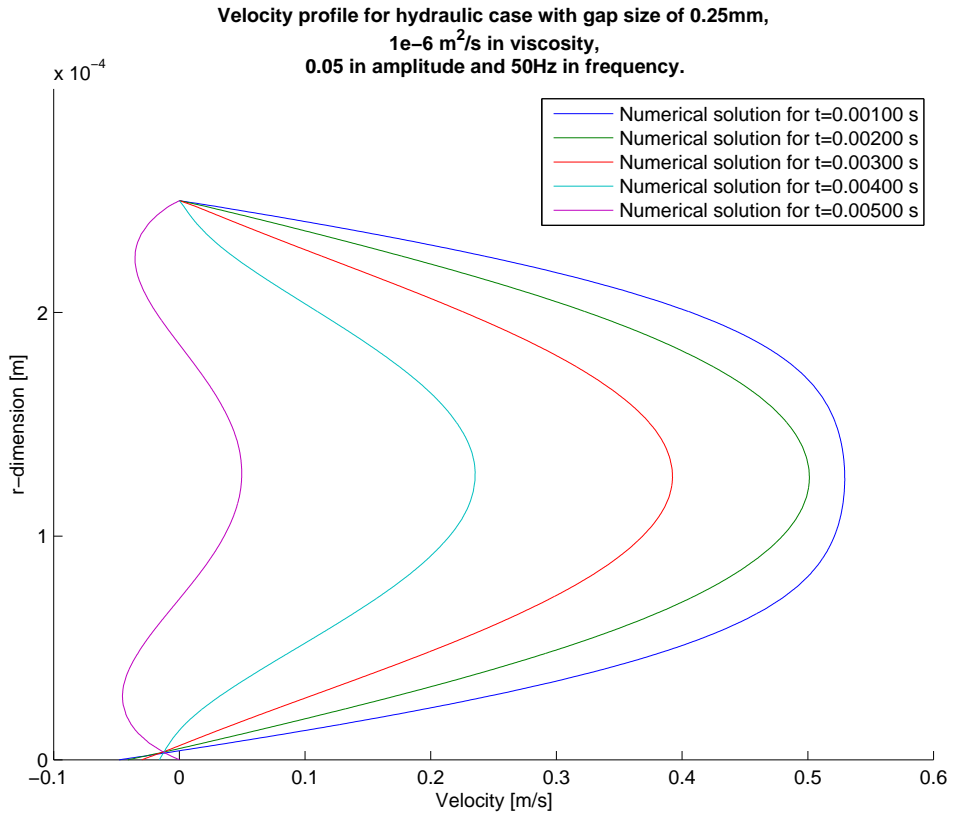


Figure 3.7: Velocity development as the cylinder is changing direction for the hydraulic piston case. B-value of 1.767

The velocity profile for the entire simulation interval is plotted in figure 3.8 on the following page, and there is nothing in this simulation that suggests that something is wrong.

No further analysis was done on this verification-case, but the

mathematical foundation along with the theoretical foundation are valid for both this case and dynamic annular cylinder case.

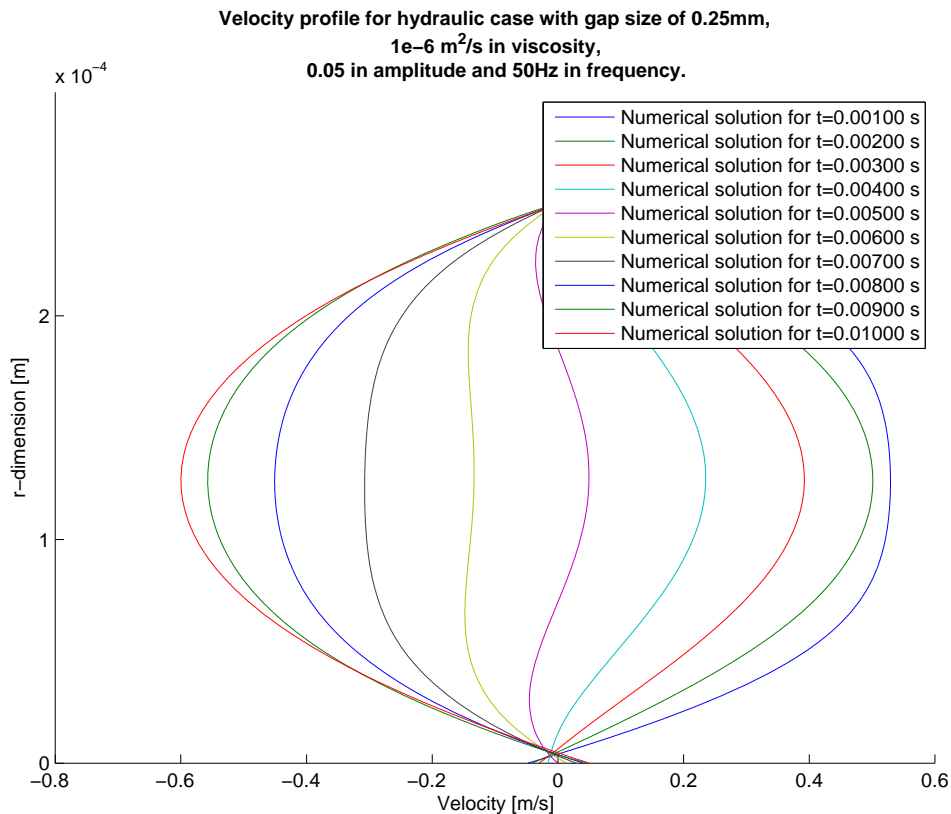


Figure 3.8: Overall velocity development for the hydraulic piston case.

3.2 Simulations

The goals for the simulation of this thesis is to develop a fully functional dynamic mesh model with periodic boundary conditions and expand from that to take into account what type of forces the cylinder will be subjected to. The long term goal is to get a realistic dynamic model that can be used to compare the results from the physical test bench.

In order to use a dynamic mesh, a solver that supports this type

of feature has to be chosen. Table table 3.1 lists all solvers that have dynamic mesh capabilities in OpenFOAM at this time.

Name	Description
pimpleDyMFoam	Transient solver for incompressible, flow of Newtonian fluids on a moving mesh using the PIMPLE (merged PISO-SIMPLE) algorithm.
rhoCentralDyMFoam	Density-based compressible flow solver based on central-upwind schemes of Kurganov and Tadmor with moving mesh capability and turbulence modelling.
sonicDyMFoam	Transient solver for trans-sonic/supersonic, laminar or turbulent flow of a compressible gas with mesh motion.
interDyMFoam	Solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach, with optional mesh motion and mesh topology changes including adaptive remeshing..
icoUncoupledKinematicParcelDyMFoam	Transient solver for the passive transport of a single kinematic particle cloud.

Table 3.1: Table with all currently available dynamic mesh solvers for OpenFOAM.

OpenFOAM has five dynamic mesh solvers and since this is a laminar case with assumed incompressible fluid, only two solvers are appropriate for this type of simulation. *interDyMFoam* is a solver for multiphase flow, which makes *pimpleDyMFoam* the only suitable solver for this type of simulation. [8, p 83-87]

This thesis will focus on a dynamic mesh model with motion in one direction. The model can easily be expanded to more advanced motion in all three vector directions, and it is also possible to expand the model to a six-degree of freedom model. This is too advanced for the initial

study, but it may be done in a future analysis of this geometry.

3.2.1 Challenges and problems

In order to set up a dynamic simulation for this case, the need for a proper way of defining the case has been a challenge. How to make sure that the proper boundary and topological conditions are correct was one of the main concerns.

The amplitude of this system has to be small in order to keep the system stable during the whole simulation interval. If the amplitude of the inner cylinder is large, the gap-size between the cylinders gets infinitely small during the simulation, it will induce large gradients. This will play a large role on the Courant stability criteria. All other simulations will be performed with a constant amplitude and in order to find the proper amplitude, a study on the effect of amplitude changes has to be performed.

3.2.2 Geometry simulation

The first case to be simulated is the case with decreasing gap size between the cylinders. The foundation for this case can be found in [14] and is an elementary case that need to align with the new simulation case in order to verify the acceptance of the new dynamic solver. The case is with a sinusoidal acceleration input that is converted to a displacement on the inner cylinder as in equation (2.20) on page 23. Alpha values from 0.5 and up to 0.92 was chosen and the force output from this simulation case is shown in figure 3.9.

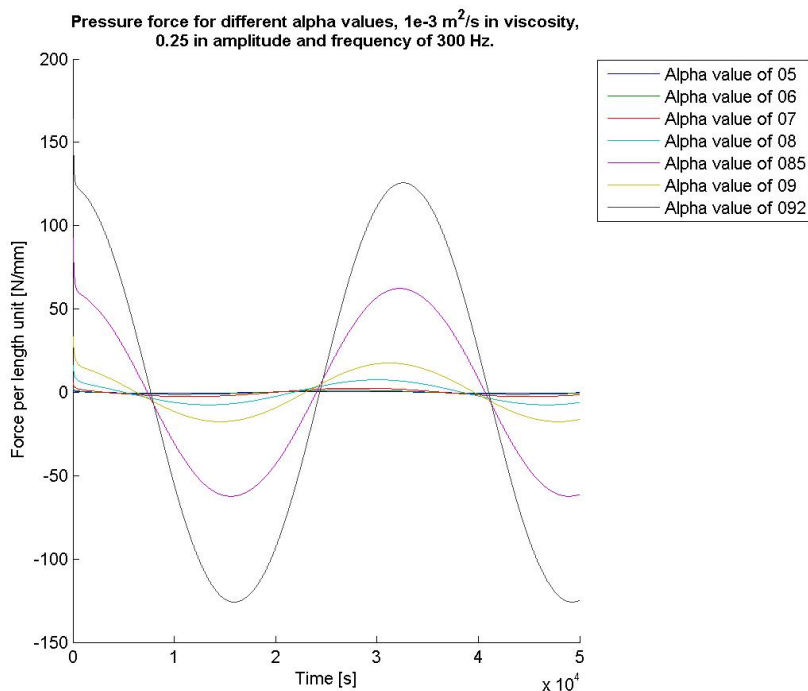


Figure 3.9: Pressure development for case with varying alpha values.

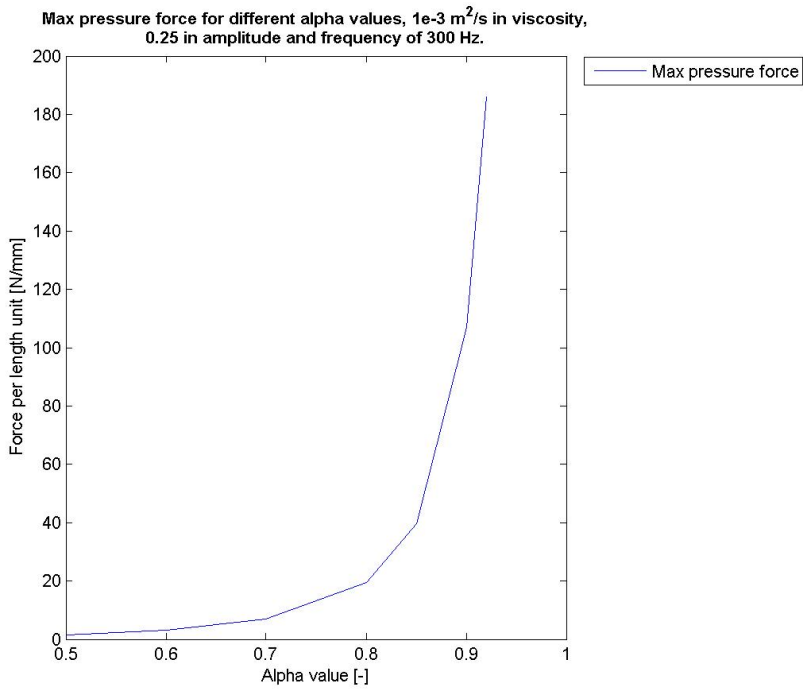


Figure 3.10: Max pressure for case with varying alpha values.

The graph in figure 3.9 on the preceding page is not comparable with the results from the project thesis and in order to compare the results, a graph that takes the pressure force as a function of the alpha value is needed. This is shown in figure 3.10 and is in agreement with the results achieved in [14].

3.2.3 Mesh

The mesh for this case is based on the mesh developed in the project thesis. [14] The mesh is changed to work in accordance with the moving mesh method. The *blockMeshDict* file for the mesh may be seen in attachment A.8.

The difference between the mesh at initial condition and in compressed condition can be seen in pictures figure 3.11 and figure 3.12 on the next page. This mesh is based on the uniform diffusivity method where the mesh compresses equally over the entire moving range.

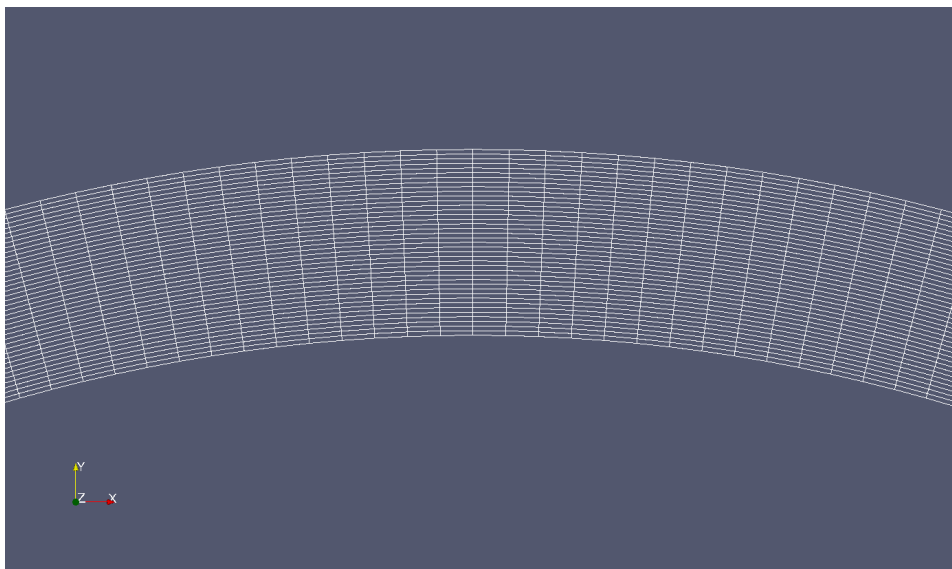


Figure 3.11: Cylindrical moving mesh at initial state along y-axis.

An important factor when using the diffusivity method is that the mesh skewness of the cells are kept within a reasonable limit during the entire simulation. In figure 3.12 on the following page you can see that the skewness of the mesh cells increases as the mesh compresses, and if the amplitude of the mesh motion is too high the mesh will become unstable and not usable. [24]

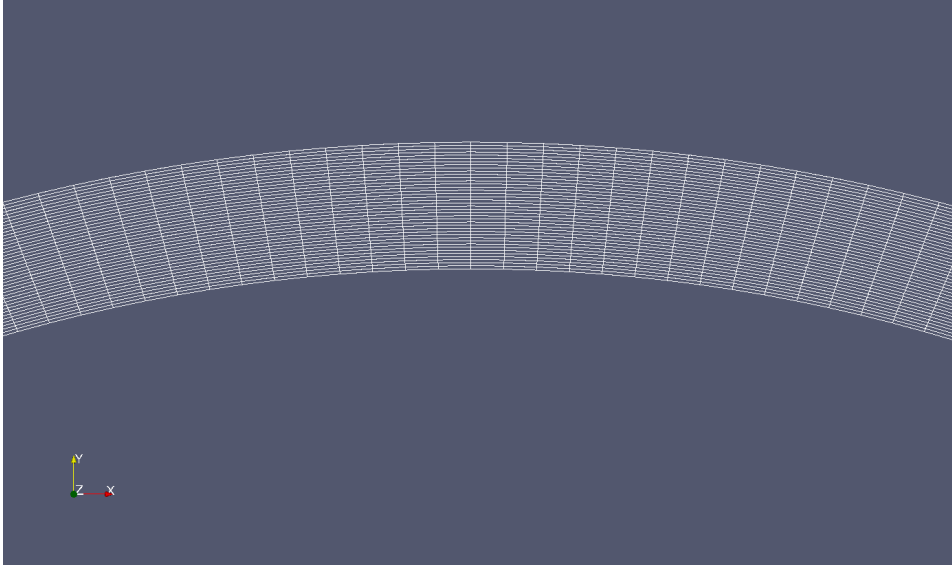


Figure 3.12: Cylindrical moving mesh at compressed state along y-axis.

Algebraic mesh motion is only suited for prescribed mesh motion and cases of regular oscillating geometries. [20]. The initial simulation case is based on this moving mesh method. More advanced methods such as *topological changing methods* can be used in order to optimize the case for higher amplitudes and lower simulation time.

In figure 3.13 on the next page and figure 3.14 on the facing page the difference between the mesh along the x-axis is clearly visible. The mesh along the x-axis is the limiting mesh cells for this case. There is no specific solution on how large the mesh skewness can be for dynamic mesh motion cases. [22] The mesh itself stays valid even though the simulation case crashes. It varies from case to case and preliminary tests need to be performed in order to find a suitable range for the amplitude of the periodic motion.

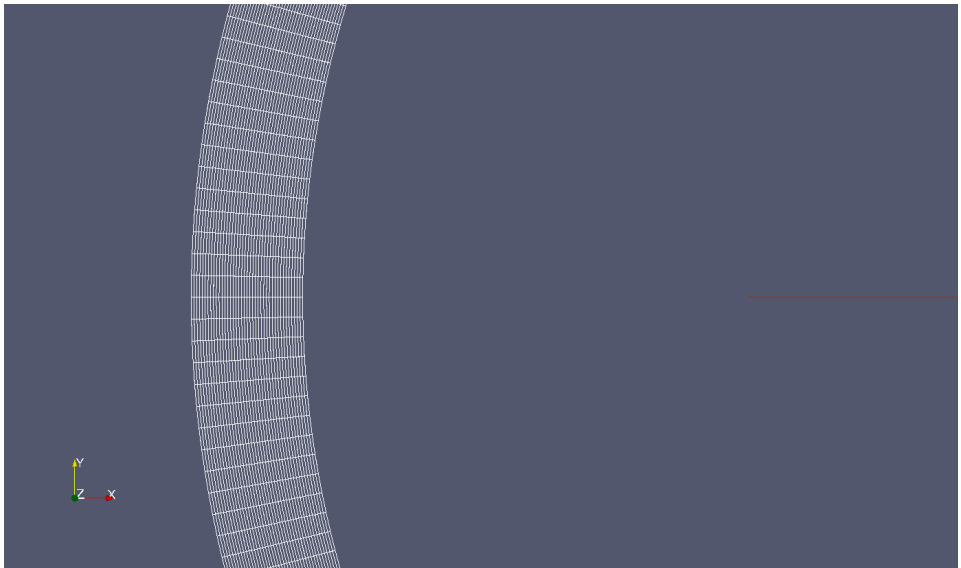


Figure 3.13: Cylindrical moving mesh at initial state along x-axis.

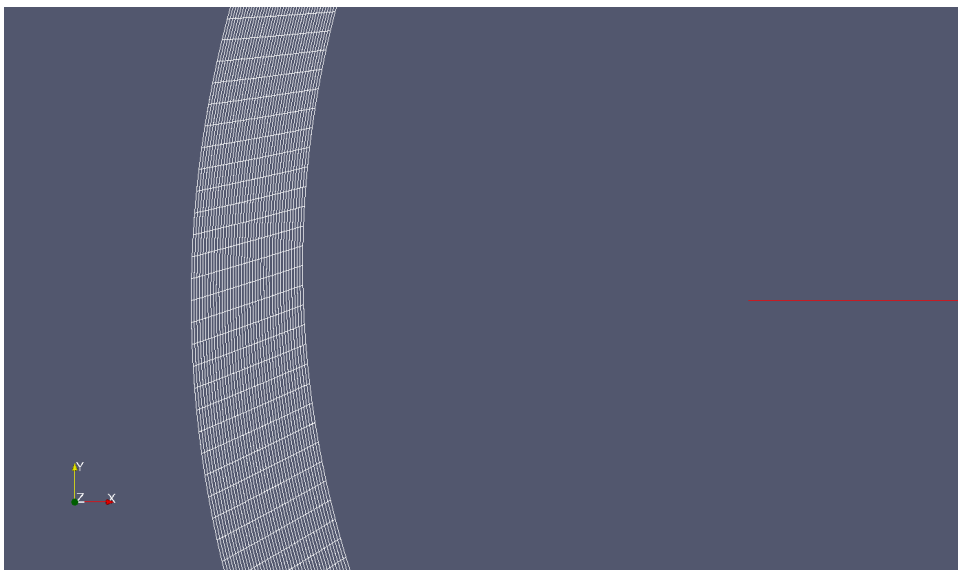


Figure 3.14: Cylindrical moving mesh at compressed state along x-axis.

Initial simulations will be performed with a low velocity amplitude in order to keep the simulation case stable over the entire simulation interval.

In order to create a better mesh for this simulation case, there is a need to know more about the velocity gradients during the simulation. With this knowledge a more specialized mesh may be used and by that increasing the accuracy and reducing the simulation time.

3.2.4 Velocity and pressure gradient distribution

An important parameter to be aware of is the velocity or pressure gradient. Since velocity and pressure are directly coupled it is only necessary to know about one of them. By knowing how the gradients are during the simulation extra precaution can be used on the mesh where the gradients are large. The gradient will always be zero along the y-axis for this case. This is due to the fact that there is symmetry along the y-axis for this case, since the oscillating motion is set to move along the y-axis. The symmetry will be along the vector of the oscillating motion in all cases.

The expression for the pressure gradient is as follows:

$$\frac{\partial p}{\partial \theta} = -\frac{\mu}{K_\alpha} \left(\frac{r_1/r_i}{\gamma - 1} + K_\beta \right) v(t) * \cos(\theta) \quad (3.54)$$

Where the velocity component $v(t)$ is from equation (2.17) on page 23 and the other variables $K_\alpha, K_\beta, \gamma$ and r_i is from the chapter with the mathematical model. A plot of the analytical solution for the pressure gradient is given in figure 3.15 on the facing page.

Analytical solution for pressure gradient, Alpha = $0.9 \cdot 10^{-3} \text{ m}^2/\text{s}$ in viscosity, 0.25 in amplitude and frequency of 300 Hz.

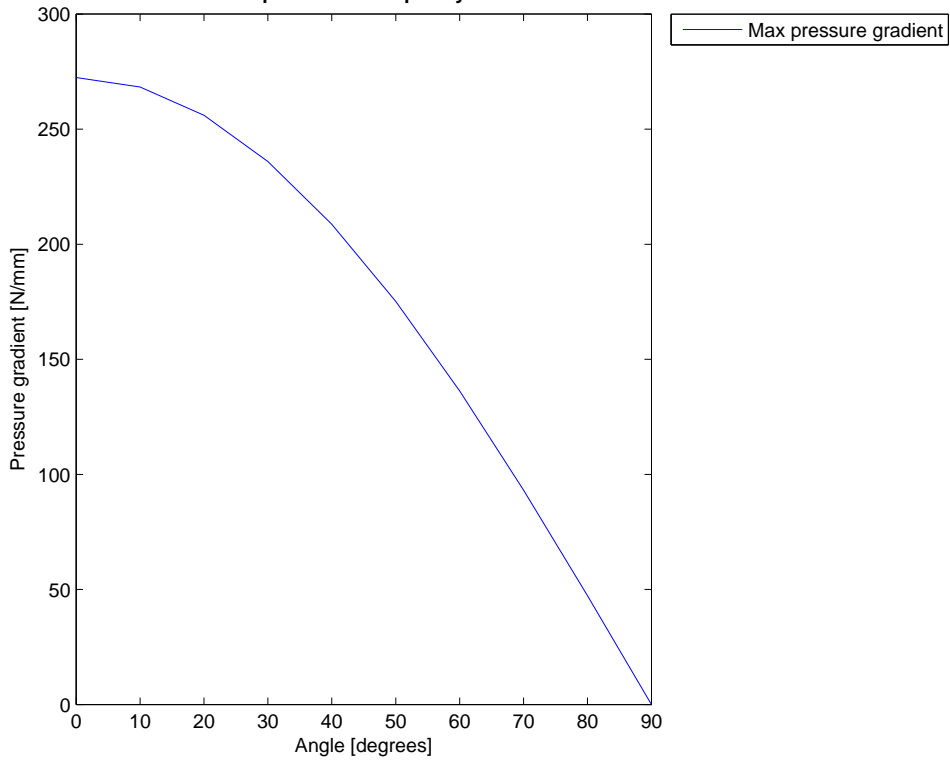


Figure 3.15: Analytical solution for max pressure gradient as a function of angle.

As can be seen in figure 3.15 the pressure gradient is largest along the x-axis and minimum along the the y-axis. This is as expected due to the symmetry along the y-axis.

In order to fully understand the pressure gradient of the simulation case there is a need to know how it varies over time. In figure 3.16 on the following page the pressure gradient is plotted over time. It can be seen in this plot that the pressure gradient is increasing as the inner cylinder is moving closer towards the outer cylinder. When the inner cylinder is moving down again and crossing the x-axis all gradients are zero as the first time period is over.

Analytical solution for pressure gradient, Alpha = 0.9, $1e-3$ m²/s in viscosity, 0.25 in amplitude and frequency of 300 Hz.

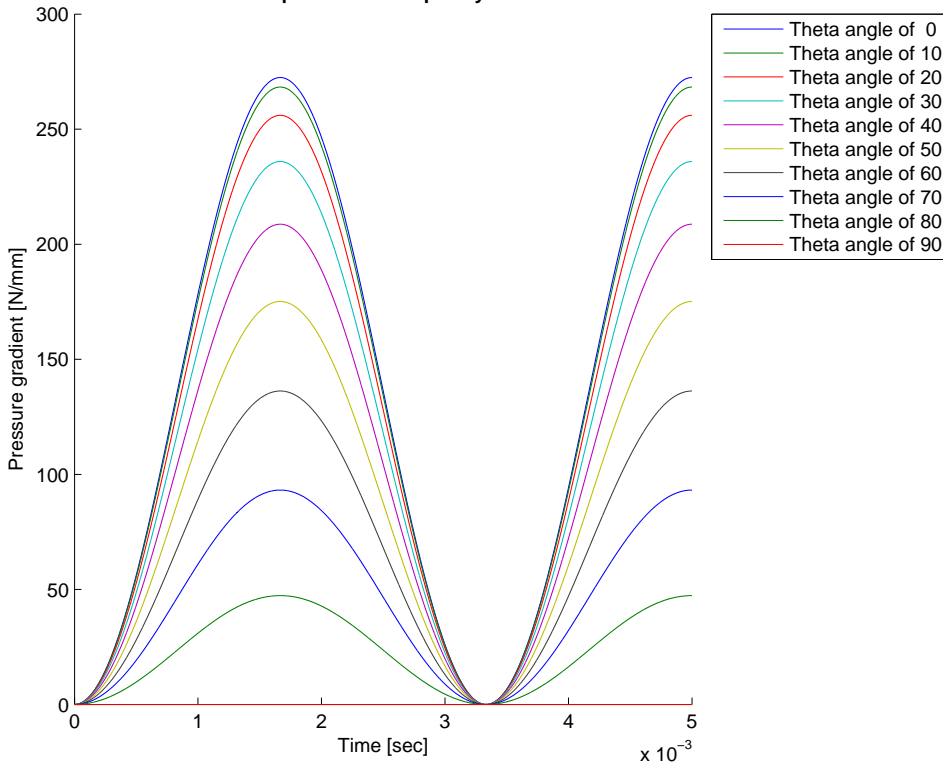


Figure 3.16: Analytical solution for pressure gradient as a function of time.

This analysis give insight in how the pressure gradient acts for this case and this can be used to better adapt the mesh.

3.2.5 Stability of simulations

In order to keep the simulation stable the Courant number has to be below 1 as stated in equation (2.15) on page 22. Another important criterion is the sampling rate of the simulations. In order to sample a dynamic motion at 500Hz at a decent quality at least 20 samples per time period is needed. That gives a Δt of $1e-4$. This was set as the initial time step for the simulations. The Courant number was not below the stability criteria with this sampling rate and a smaller time

step was chosen. The time step was decreased by an order of magnitude until all simulations were stable. At Δt of $1e^{-7}$ all simulations except for the ones at 50 and 100Hz were stable. Most of the simulations showed good signs of stability and the Courant number was well within the stability criteria. This can be seen in figure 3.17.

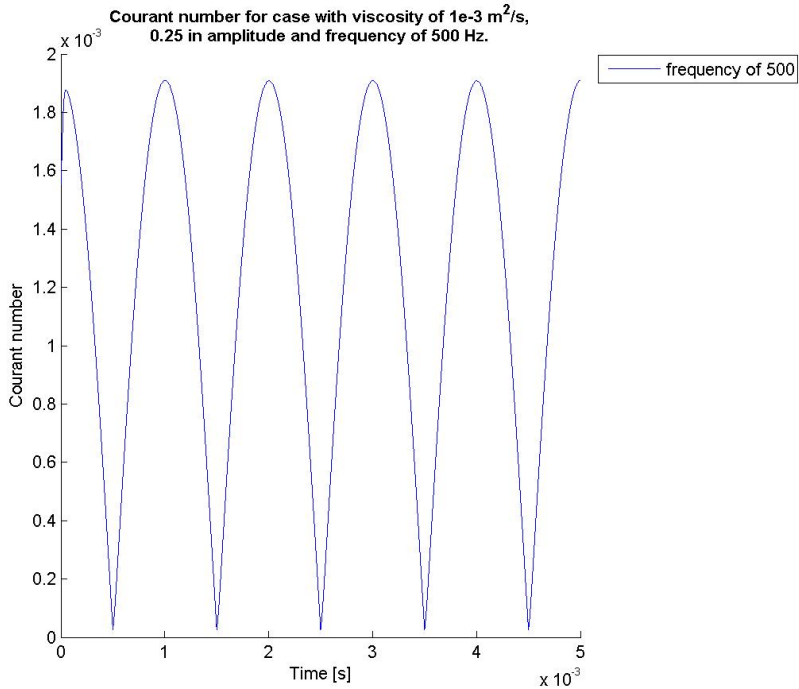


Figure 3.17: Courant number during a dynamic mesh simulation.

Another important criterion was to see how much the Courant number changed with different frequency and viscosity. The change in courant number based on frequency is shown in figure 3.18 on the next page and it can be seen that there is no noticeable change in the maximum Courant number.

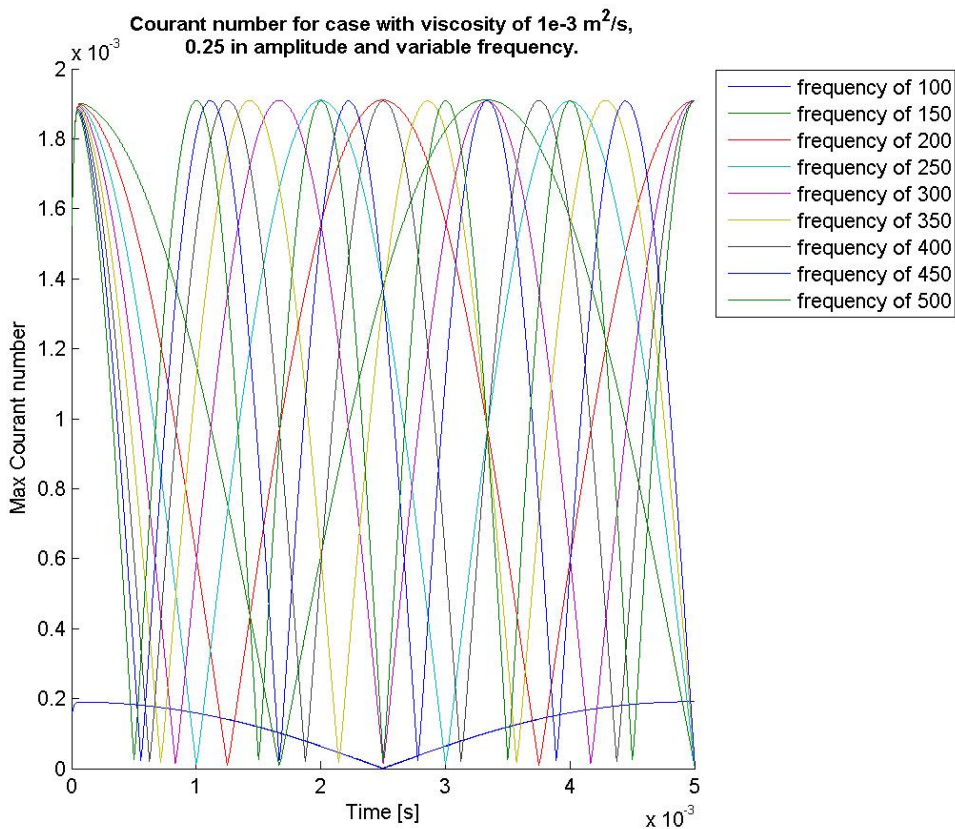


Figure 3.18: Courant number for different frequencies at one viscosity value.

Notice the lower Courant curve for the frequency of 100Hz. This is due to the fact that this simulation is done with a Δt of an order of magnitude below the other simulations in order to keep that simulation case stable.

The same study was performed for the viscosity. For variable viscosity there is a small change in the beginning of the simulation, but there is no change in the maximum Courant number for viscosity changes as well as frequency changes. This rule out the factor of the Courant number being an issue for the different simulations.

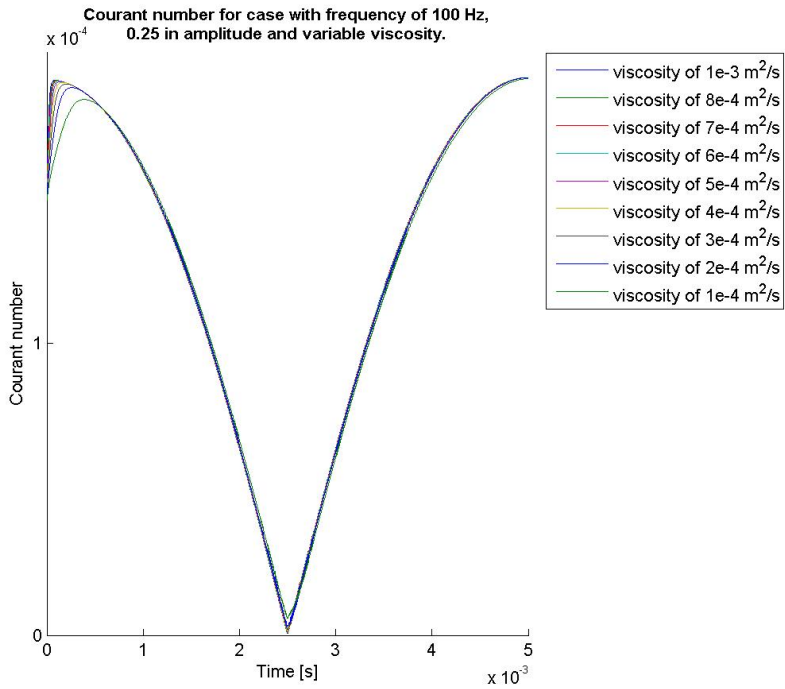


Figure 3.19: Courant number for different viscosities at one frequency value.

This analysis means that the lack of stability for the low frequency inputs must be due to another factor.

3.2.6 Richardson’s annular effect

The entrance length theory has been superseded by the *Richardson’s annular effect* and it is interesting to see how this near-wall velocity overshoot affects the pressure force distribution and thereby the damping of system. Also it is natural to expect that the viscous forces of the system increases as the velocity overshoot implies that the velocity gradients increase near the wall.

By checking all the simulations done in this thesis against this requirement there is a clear line where the simulations are above this value. This can be seen in figure 3.20.

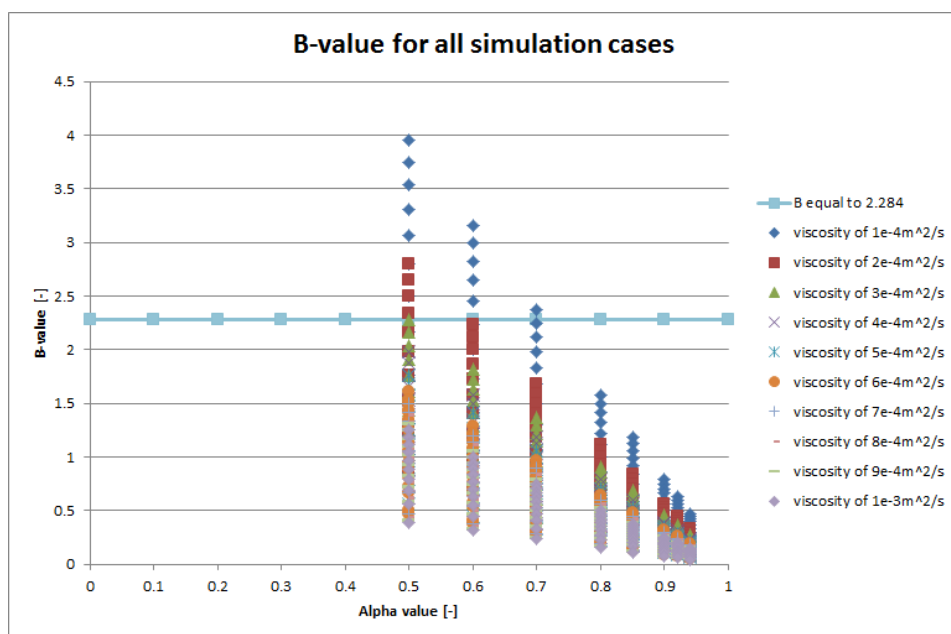


Figure 3.20: Chart of the B-value for all simulations done in this thesis.

The line in the figure 3.20 indicate that only a small sample of the simulated plots are over this value. All cases are for low α values.

It is interesting to check all cases to see exactly where the transition from regular parabolic velocity profile to the profile with the near-wall

velocity overshoot is.

Two cases with different B values has been used to see if this effect was present. The first case is a case with a B-value of 2.45 and the intermediate velocity profiles for this case can be seen in figure 3.21. Intermediate means that the velocity profiles are plotted for the time interval where the inner cylinder turns within the system. The intermediate velocity profiles for all other cases are available in appendix B.

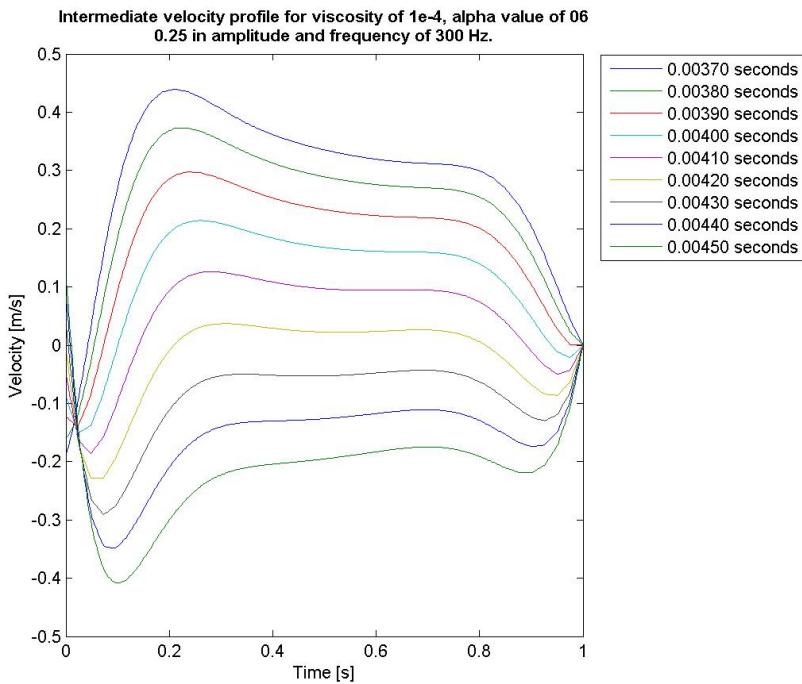


Figure 3.21: Plot showing the velocity profile for case with B-value of 2.45

The other case is a case with a B-value of 0.15. This case shows no signs of the near-wall velocity overshoot. This is in accordance with the theory and figure 3.22 on the next page shows that this is clearly a parabolic flow. These two figures are only a small part of all the cases that were tested. All cases can be seen in part two of the attachments.

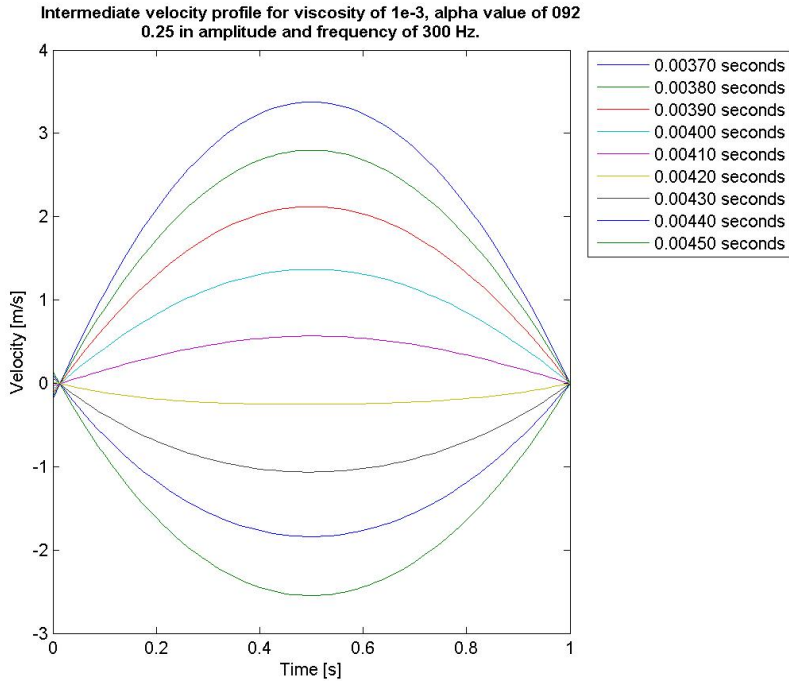


Figure 3.22: Plot showing the velocity profile for case with B-value of 0.15

3.2.7 Frequency simulation

The frequency range for this simulation case is set to be from 50 to 500Hz. This is the area where the damper is used and due to the limited timespan of this assignment the frequency span is set this way. The frequencies that are selected are at a uniform interval of 50Hz between each sample.

Max velocity as a function of frequency is plotted in figure 3.23 on the facing page and it can be seen that as the viscosity increases, the value for max velocity converges for most of the frequencies.

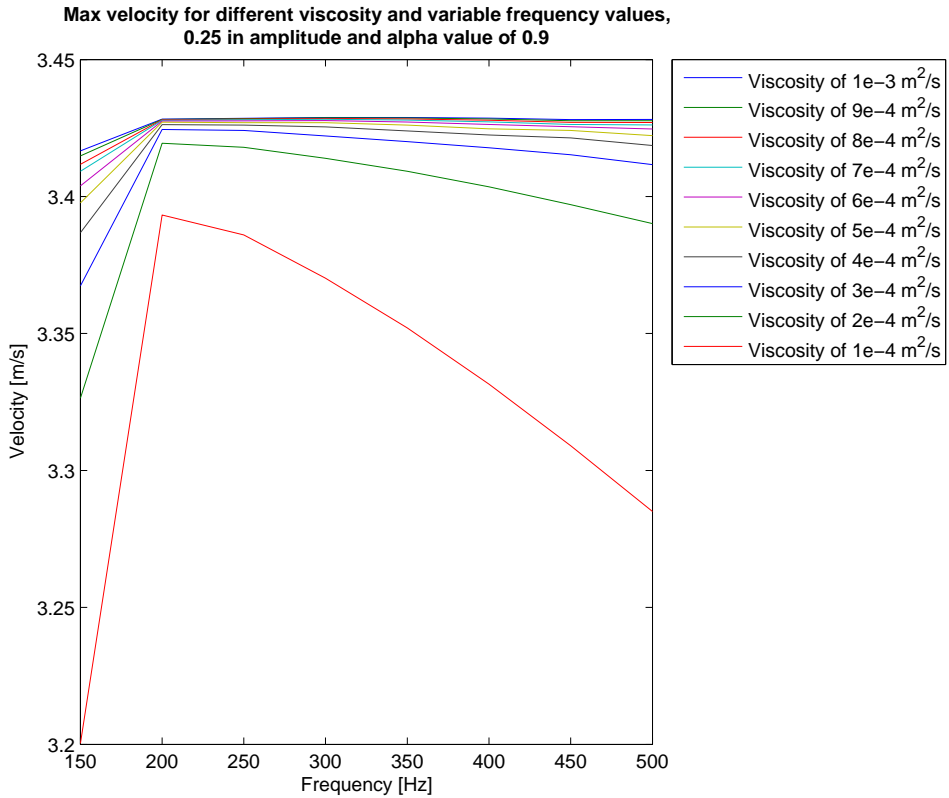


Figure 3.23: Max velocity as a function of frequency and viscosity.

The max velocity appear to have a decreasing trend for frequencies lower than 150Hz. This applies for all plotted viscosities. This is something that could be researched in more detail, but trouble with low frequencies has restricted this study.

Some of the frequencies have a decreasing trend for frequencies above 200Hz as well. This applies for the simulations with small viscosities and suggests that a further study on high frequency effects for low viscosity fluids should be performed. That type of case will have a large kinetic Reynolds number and be on the verge of the turbulent area. No turbulent effects have been studied in this thesis, so no reason to expand further on this effect.

3.2.8 Amplitude simulations

The amplitude of the input response was not the primary variable to conduct experiments on. This is because changes in the amplitude of the system should act linearly at least for a span of amplitudes. So long as the inner cylinder does not get close to the outer cylinder, the value of the amplitude should be trivial. The amplitude has been kept constant for all other simulations. Research on the effects of larges amplitudes giving small gap-size effects could have been performed, but this is work that should be conducted when the model has been properly tested and verified.

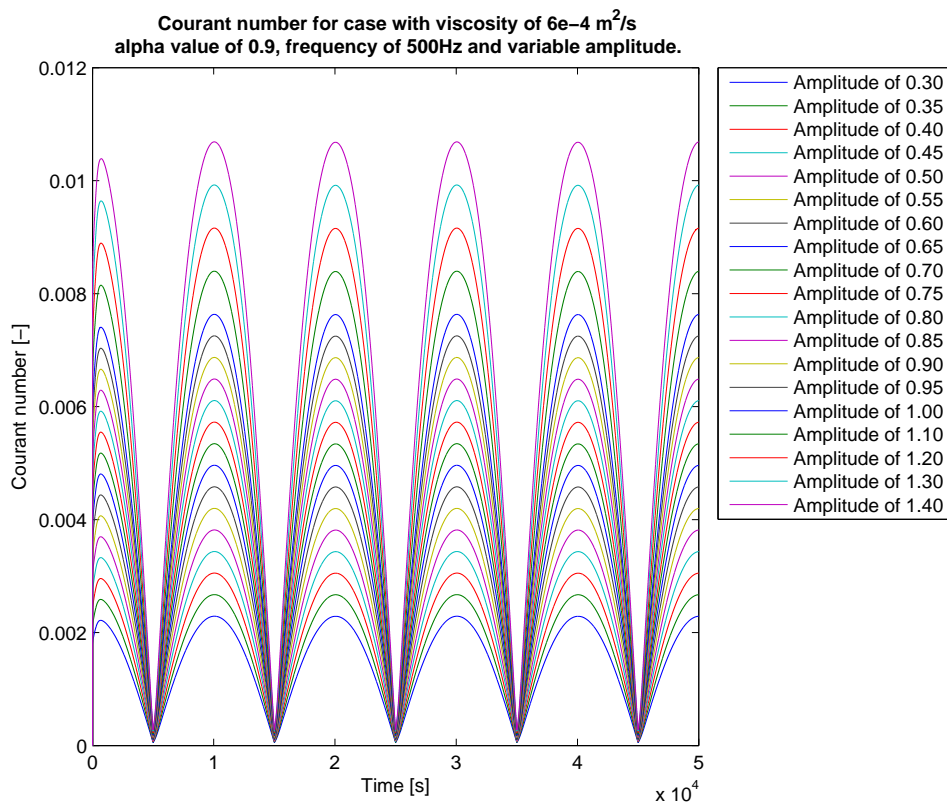


Figure 3.24: Courant number for simulations with different amplitudes.

In order to control the stability area of the simulation model a study

of the Courant number for changing amplitudes was conducted.

The Courant numbers for different amplitudes has been summarised and are plotted in figure 3.24 on the preceding page. It can be seen that the amplitude analysis stops at 1.40 in this figure, and that is because of unstabilities at higher amplitudes. In order to stabilise the simulation the timestep had to be decreased so much, that the extra simulation time due to this, was not feasible to work further on.

For this amplitude range the Courant number is linear. It can be seen in figure 3.25.

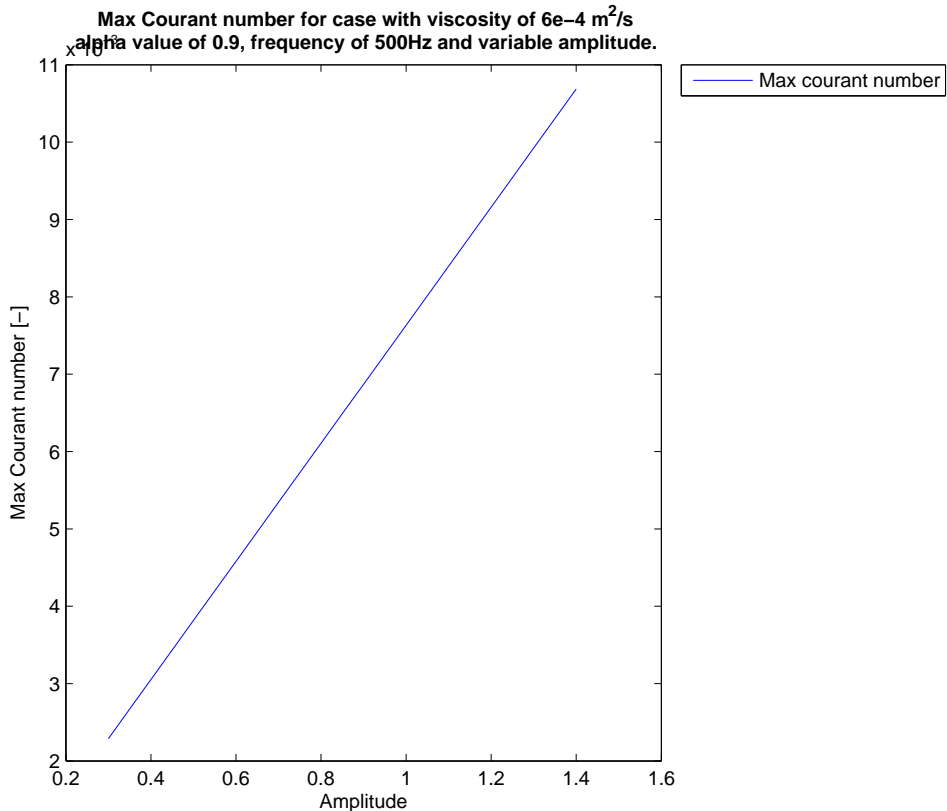


Figure 3.25: Max Courant number plotted as a function of amplitude.

The forces on the cylinder were also plotted against the amplitude and as expected they are linearly increasing with increasing amplitude as can be seen in figure 3.26 and figure 3.27 on the facing page. The pressure force is shown in the following figure.

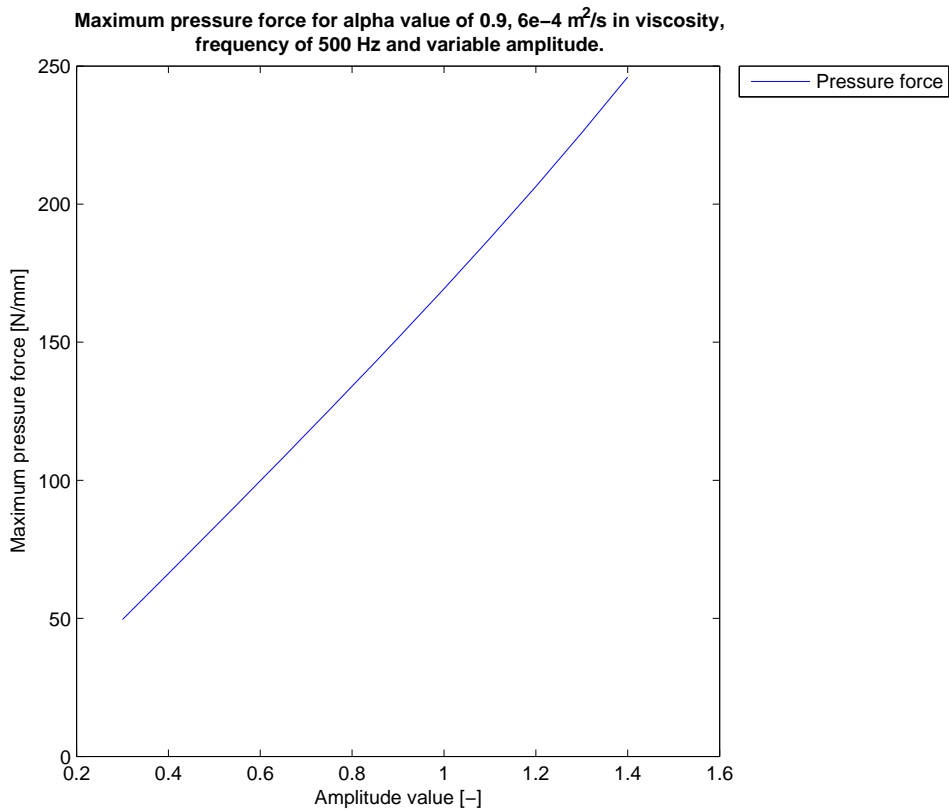


Figure 3.26: Max pressure force plotted against amplitude.

It is worth mentioning that the pressure forces are likely to increase more as the gap-size decreases, but since the amplitude study was not performed for higher amplitudes, this has not been confirmed.

The viscous force is shown in figure 3.27 on the next page and shows a linear relationship between viscous force and amplitude.

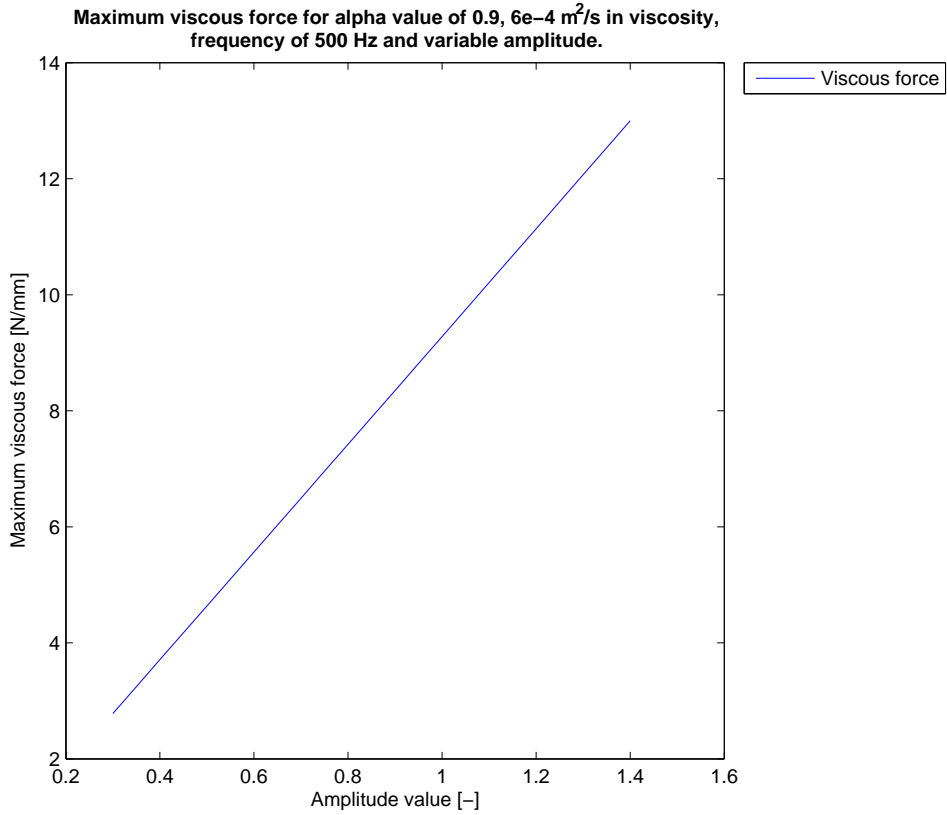


Figure 3.27: Max viscous force plotted against amplitude.

The relative force difference was also plotted for this case and it can be seen that the pressure force increases more than the viscous force for the amplitude increase. This indicates that the pressure force increases more than the viscous force as the gap-size decreases.

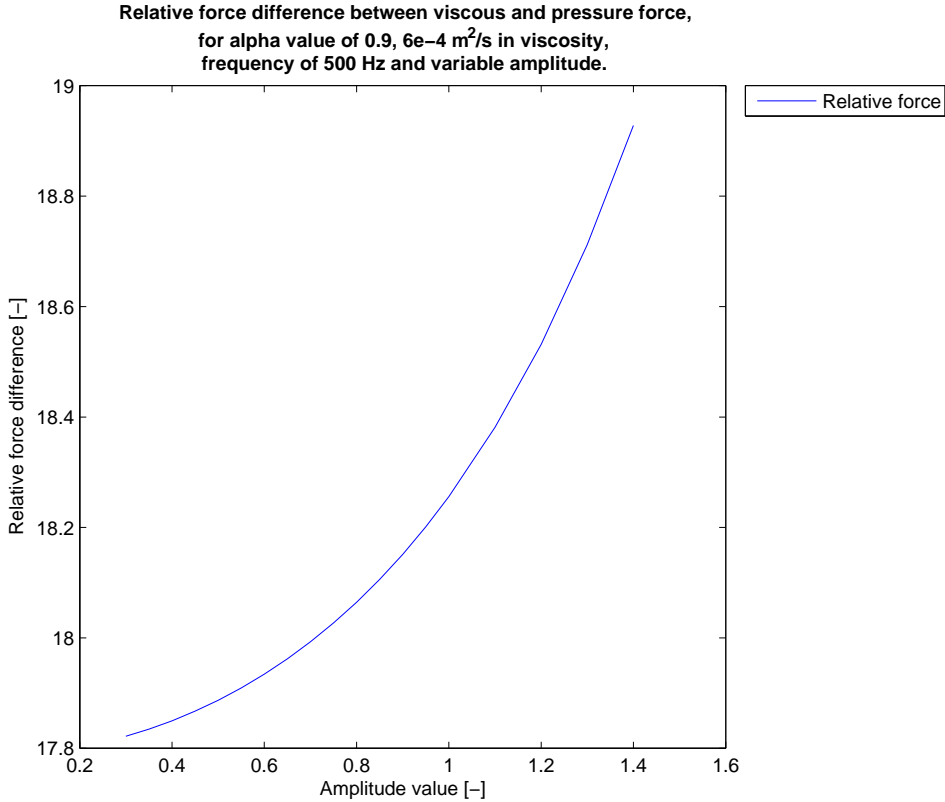


Figure 3.28: The relative force difference between pressure and viscous force plotted against amplitude.

This conducts the amplitude analysis of the system. It has not been thoroughly analysed due to the fact that this variable has not been the main focus of the thesis.

3.2.9 Velocity simulations

In order to validate the dynamic case the velocity profiles for the simulated case is compared with the calculated maximum velocity as found in the project thesis. [14]. It can be seen in figure 3.29 on the facing page that the max velocity component is in good agreement with the results achieved for the quasi-static model.

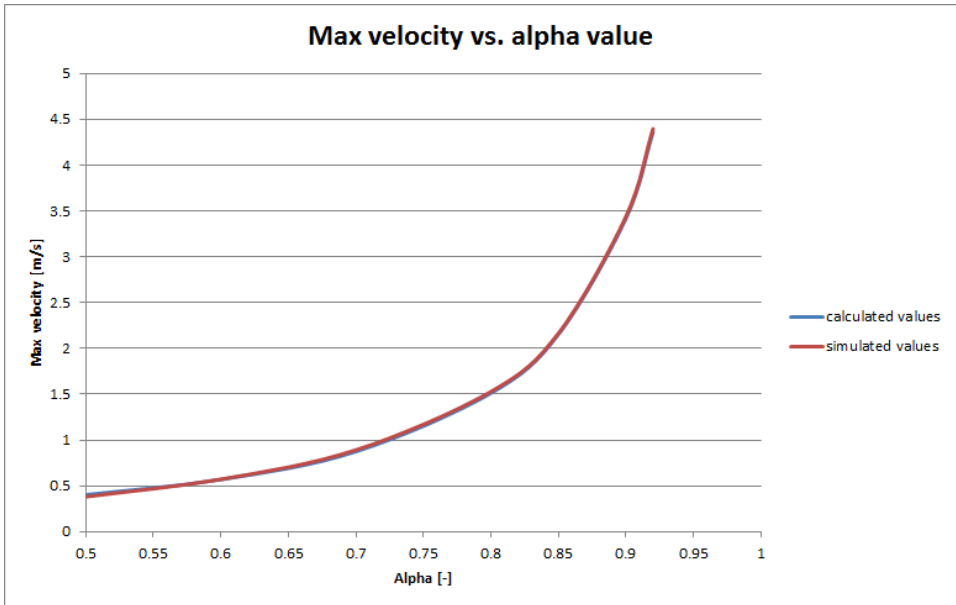


Figure 3.29: Max simulated and calculated velocity component for different alpha values.

3.3 Force output

The forces on the inner cylinder can be extracted from the simulation case by setting up a force library along with the simulation case. The force library is included in the *controlDict* file in the way as shown in lines 55 to 58 in attachment A.9. This input sets up a link to a file with the name of *forces* in the same directory. This file will specify the use of the *libforces.so* library and let you set the parameters for the force calculations. The file used for the simulations in this thesis are shown in attachment A.10.

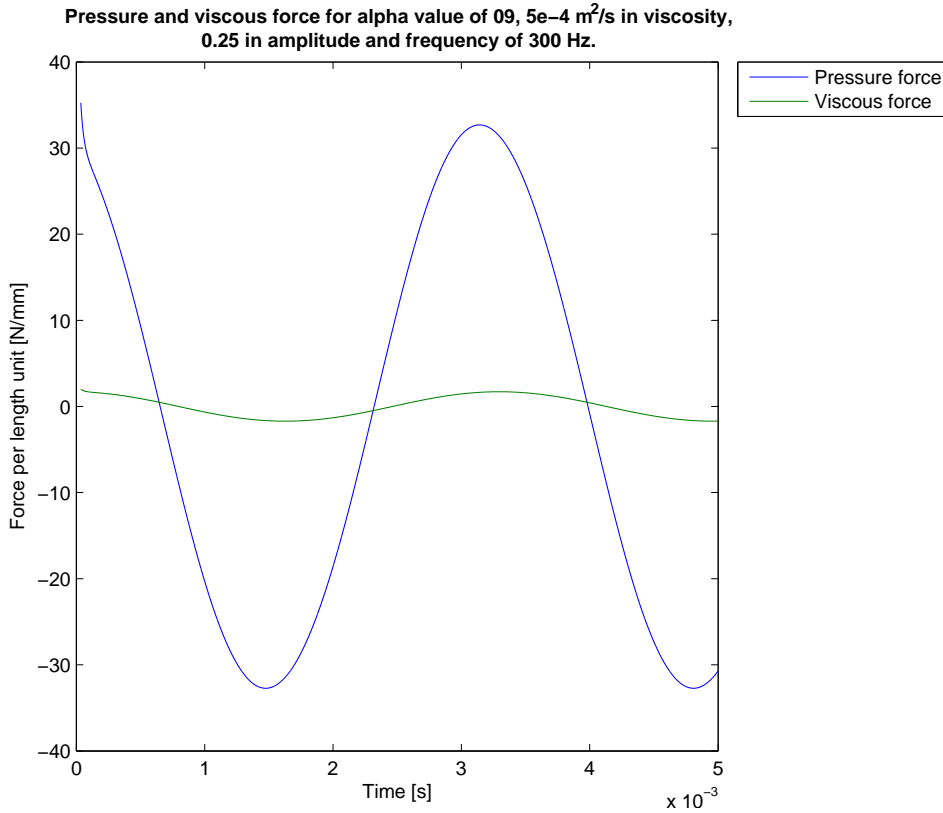


Figure 3.30: Example of force output for entire time interval for one case.

An example of the plotted force output is shown in figure 3.30 and it can be seen that the pressure is dominating the system. This force output is in agreement with the results found in the project thesis [14]. One thing that needs to be mentioned is that this graph does not take into account what happens at the beginning of the time interval. It can be seen in figure 3.30 that there is a small gap at the beginning of the simulation. The system develops a huge initial force and this effect has been ignored for this study. So all validation has been taken from the second time period of the time analysis.

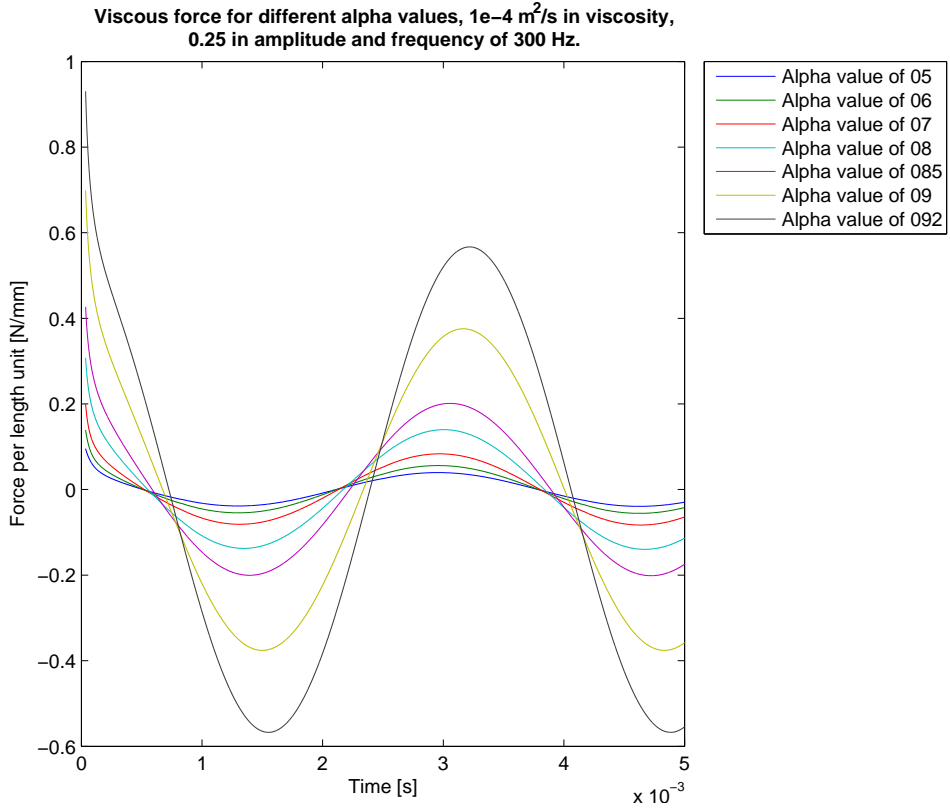


Figure 3.31: Example of force output for entire time interval for one case.

The force output for one case with variable alpha value are shown in figure 3.31.

All forces from the simulations has been collected and a summary of the maximum pressure forces as a function of both viscosity and α are shown in figure 3.32 on the following page. Here two effects can be seen. The relative force increases as the viscosity increases and the force increases as alpha gets larger. The increase in force as the viscosity increase is in agreement with equation (2.23) on page 29 that shows the linear relationship between pressure force and viscosity.

3.3.1 Pressure force

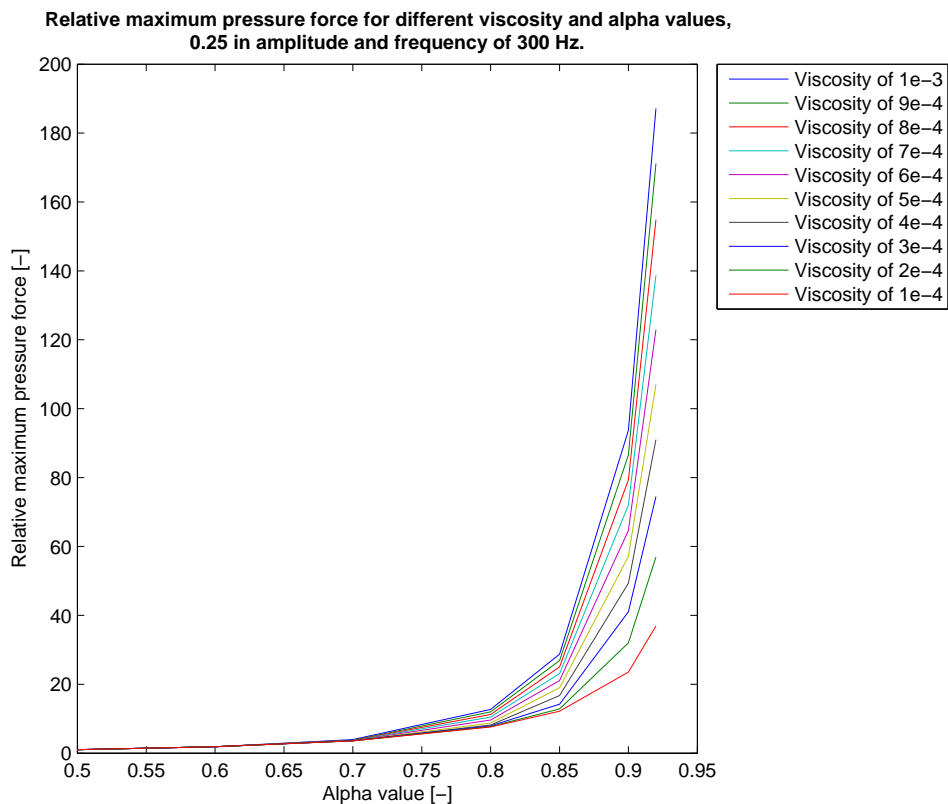


Figure 3.32: Relative maximum pressure force for variable alpha values.

The increase in pressure force is logical as the velocity magnitude increases as alpha gets larger and this imply a larger pressure force as a consequence of this velocity increase. The increase in force as a function of alpha is also in agreement with the results found in the project thesis. [14]

A study on the effect of changing frequency on the pressure force contribution was also performed. This showed that the pressure force was nearly independent of the frequency, at least for the tested frequencies. This study can be seen in figure 3.33 on the facing page.

It is worth noticing that there is an increase in the pressure force as the frequency increase and this should be studied further if possible.

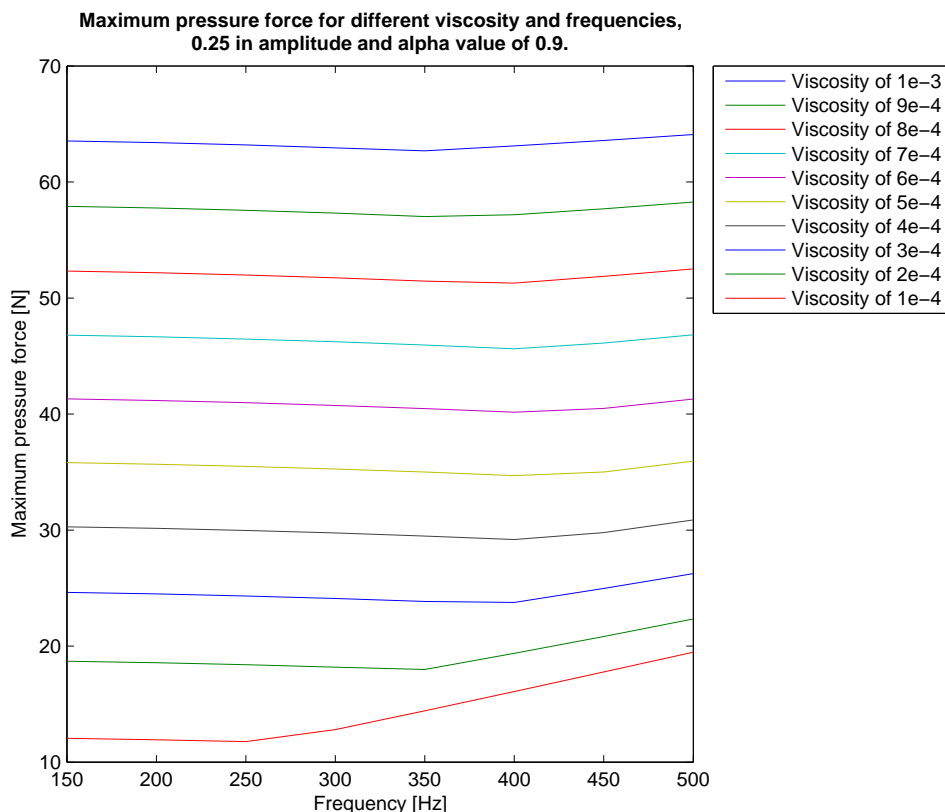


Figure 3.33: Max pressure force for variable viscosity and frequency.

3.3.2 Viscous force

The viscous force was plotted in the same manner as the pressure force. The plot has both alpha and viscosity as a variable and can be shown in figure 3.34 on the next page. The viscous force increases linearly with increasing viscosity as indicated by equation (2.26) on page 30. The viscous force increases as the velocity gradient gets larger and the velocity magnitude gets larger as alpha increases and this combination gives a large increase in viscous force as the alpha value gets close to

1.

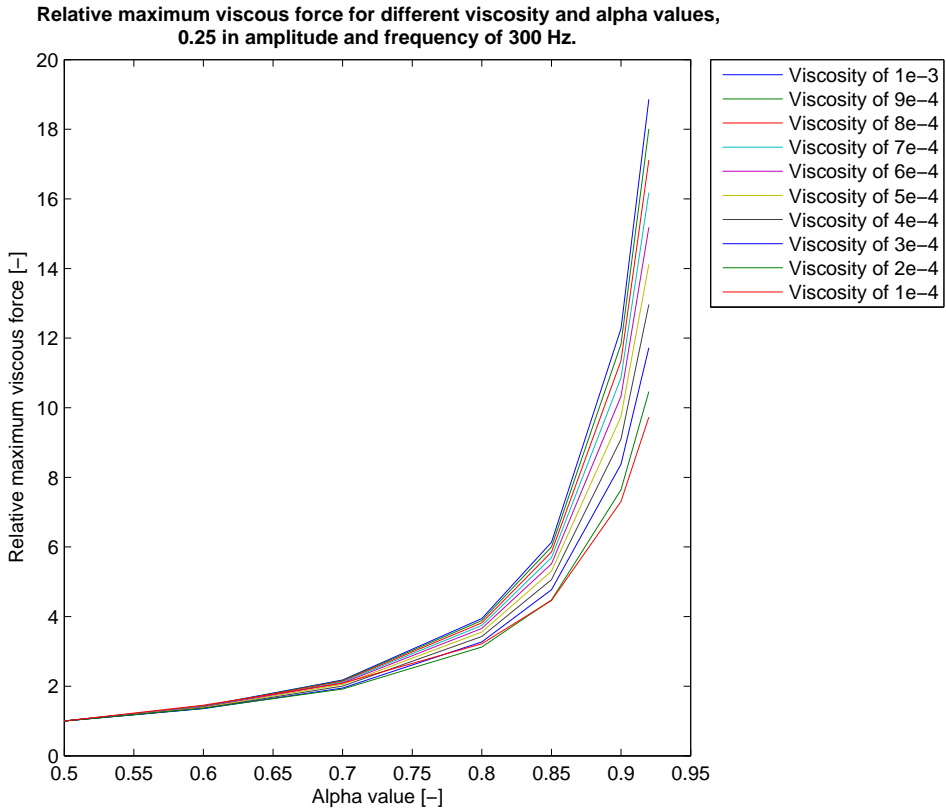


Figure 3.34: Relative maximum viscous force for variable alpha values.

The viscous force was also analysed to see how the frequency affected the viscous force. The plot for this is shown in figure 3.35 on the next page. It can be seen that the viscous force is close to independent of the frequency. There is a small drop in the force as the frequency increase, but it is very small and can be ignored.

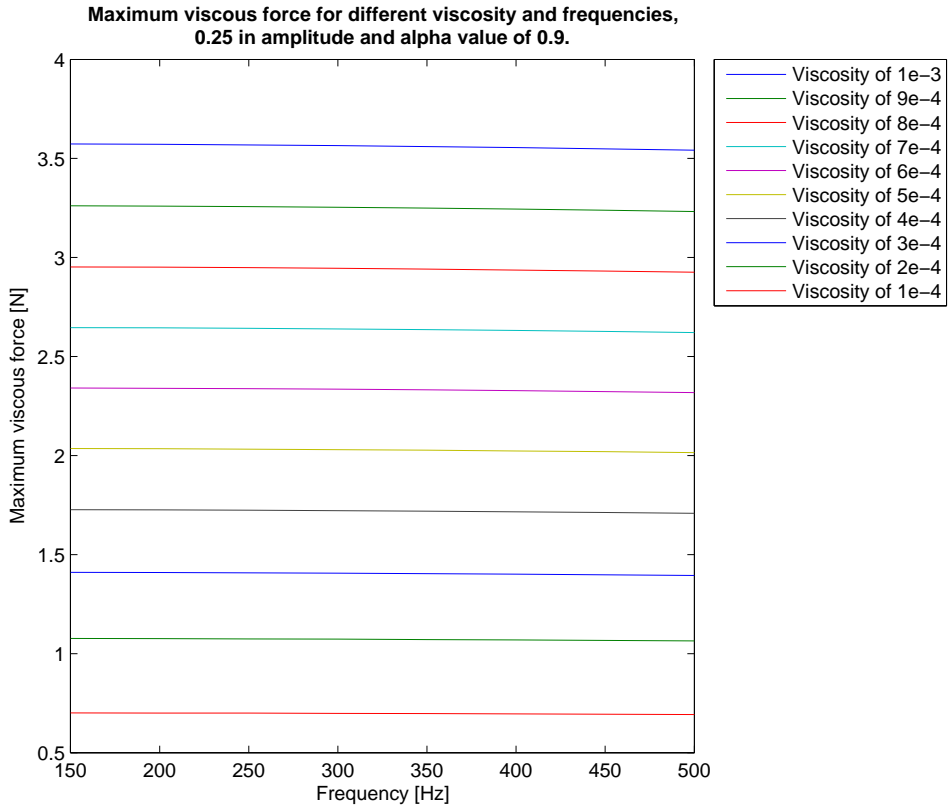


Figure 3.35: Max viscous force for variable viscosity and frequency.

3.3.3 Relative force

Now that the pressure and viscous force contribution has been identified, it is interesting to see how the relative force difference varies with the same variables. The relative force difference is based on equation (3.55). This relationship has been summarised in figure 3.36 on the following page, and it can be seen that as the alpha value increase, so does the ratio.

$$relative\ ratio = \frac{pressure\ force}{viscous\ force} \quad (3.55)$$

It can be seen from figure 3.36 that as the viscosity gets higher, the relative force difference gets smaller. This is in agreement with what one would believe. Since the viscous force is likely to increase more than the pressure force if only the viscosity is increased.

For α -values over 0.9 the relative ratio is above 15 regardless of viscosity and that gives justification to the statement that the pressure force dominates the system at high alpha values.

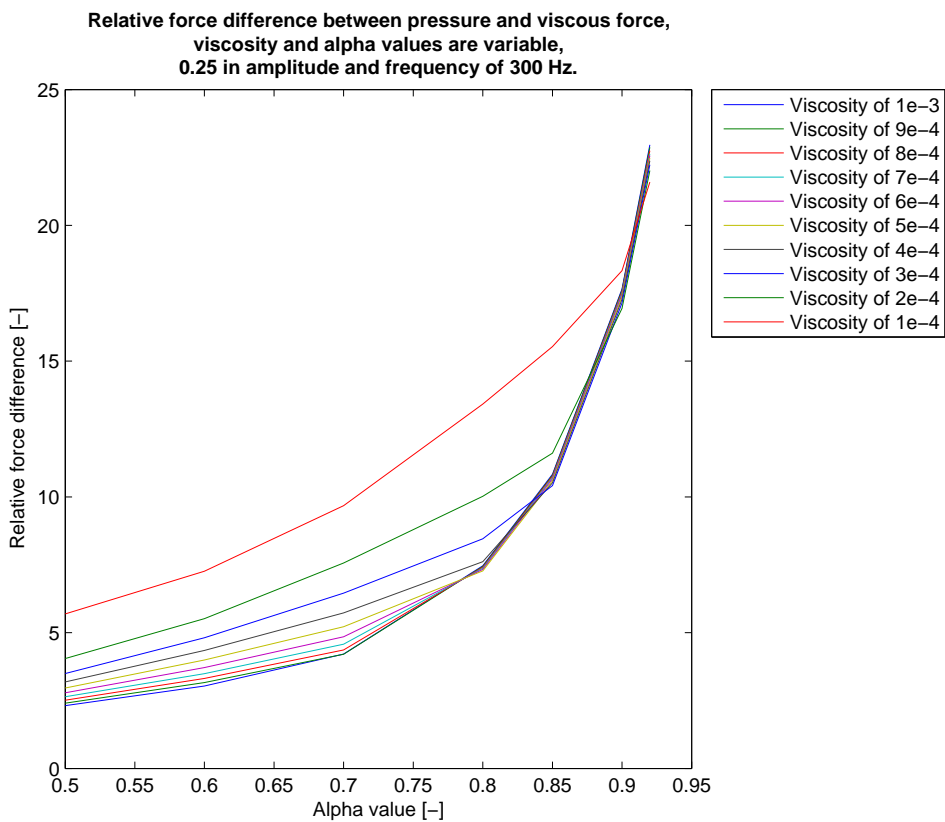


Figure 3.36: Relative force difference between pressure and viscous force for variable viscosity and alpha values.

A frequency sweep was done for the relative maximum pressure ratio

and the findings are shown in figure 3.37. From this graph it can be seen that the relative pressure force increases as the frequency increases for some of the viscosities that are tested. The increase is for fluids with low viscosity. This is the same effect that can be seen in figure 3.33 on page 77, but the effect is much easier to see in this plot. This gives even more indications that there is a phenomenon that needs to be investigated further. High frequency effects on low viscous fluids in dynamic annulus cylinders.

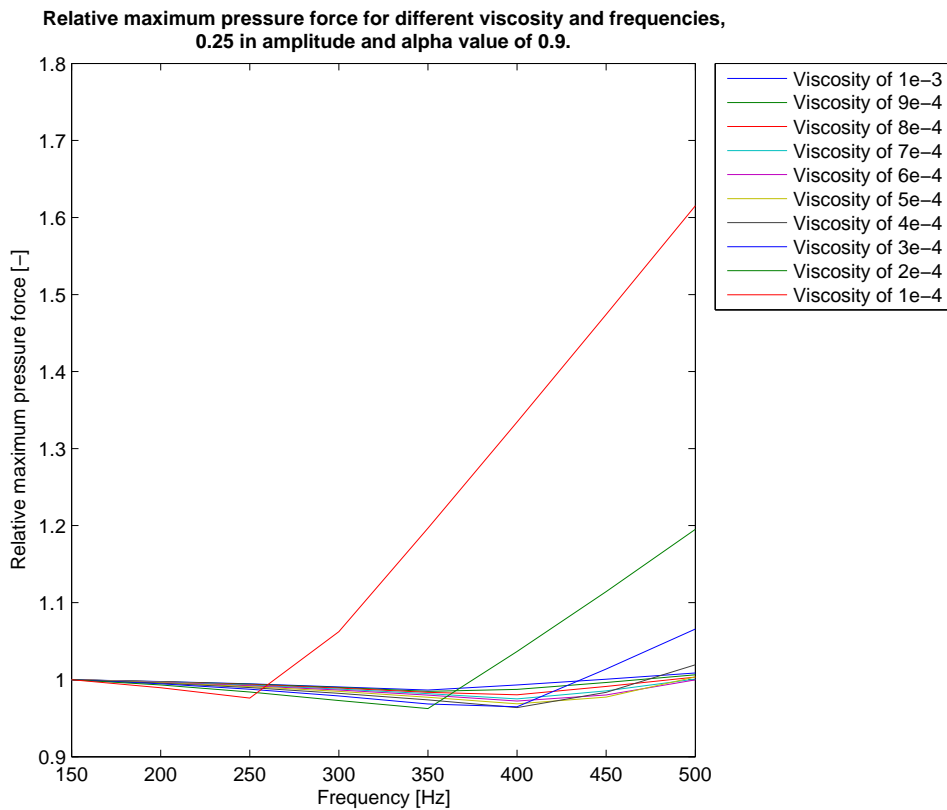


Figure 3.37: Max relative pressure force for variable viscosity and frequency.

The same sweep was performed for the viscous force in order to see if there was any high frequency effects for the viscous force as well. This plot can be seen in figure 3.38 on the following page. The trend with

the decreasing viscous force as the frequency increases is also seen in this graph. This is the same as in figure 3.35 on page 79.

The fall is less than two percent from 150 to 500Hz, so this is not worth pursuing.

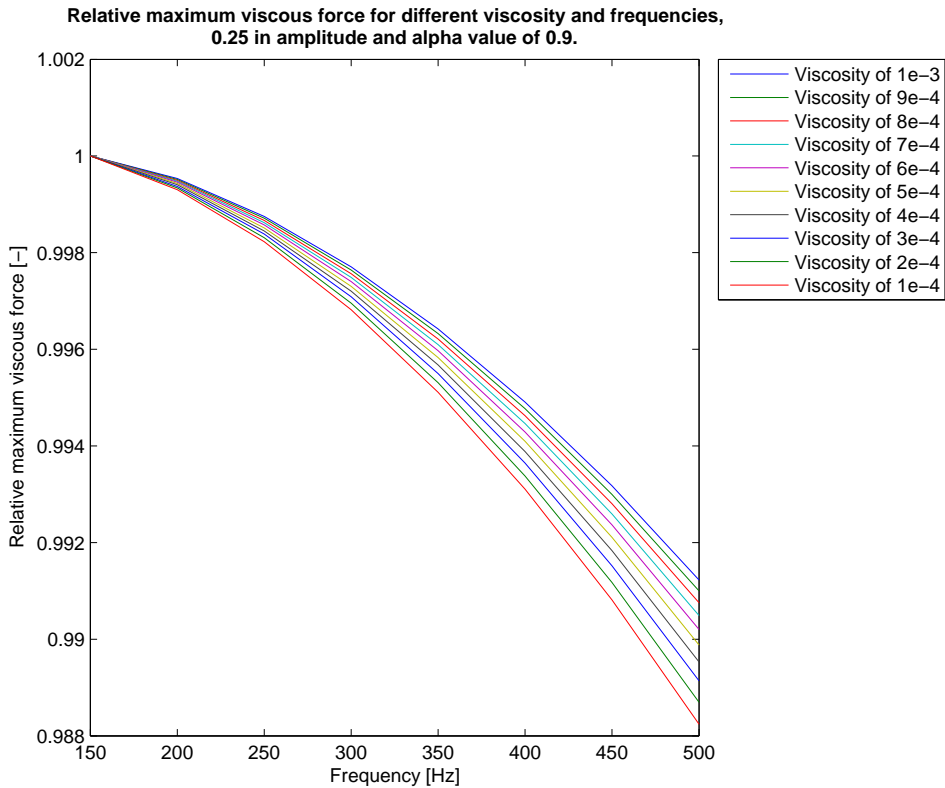


Figure 3.38: Max relative viscous force for variable viscosity and frequency.

3.4 Viscosity analysis

A small selection of viscosity values are used in this master thesis due to the limited time span of this assignment. The viscosity values are selected in accordance with the supervisor of the assignment.

The velocity of the case should be set by the geometry, but if the flow

does not have time to properly develop there should be a change in the max velocity component of the flow for different alpha values.

A simulation set-up of viscosities from $1e^{-3}$ to $1e^{-4}$ was performed on a case with 300Hz and alpha values from 0.5 to 0.92. The results are as shown in figure 3.39.

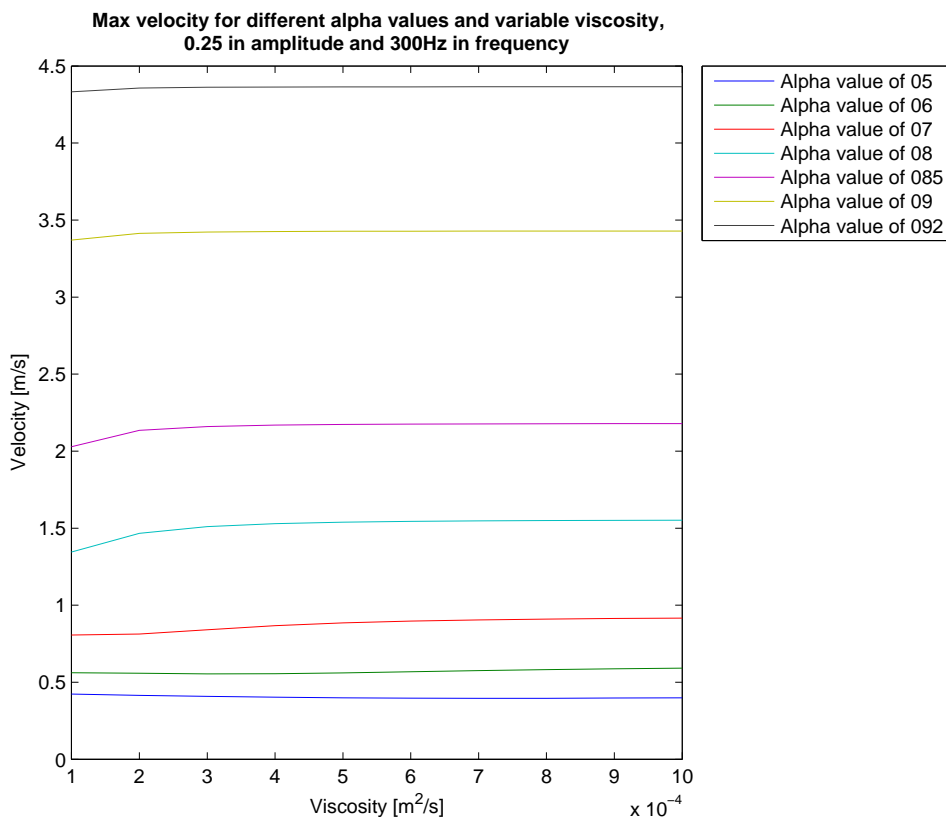


Figure 3.39: Max velocity component for different viscosities and different alpha values.

As can be seen in figure 3.39 the max velocity component is close to constant over the entire viscosity range and there is no need to assume that the velocity will not properly develop for this case.

Now that the relationship between the velocity and viscosity has been confirmed it is desired to see if all velocity profiles are in accordance

with the velocity profile found in the project thesis. [14] Figure 3.40 shows the maximum velocity for all different viscosities.

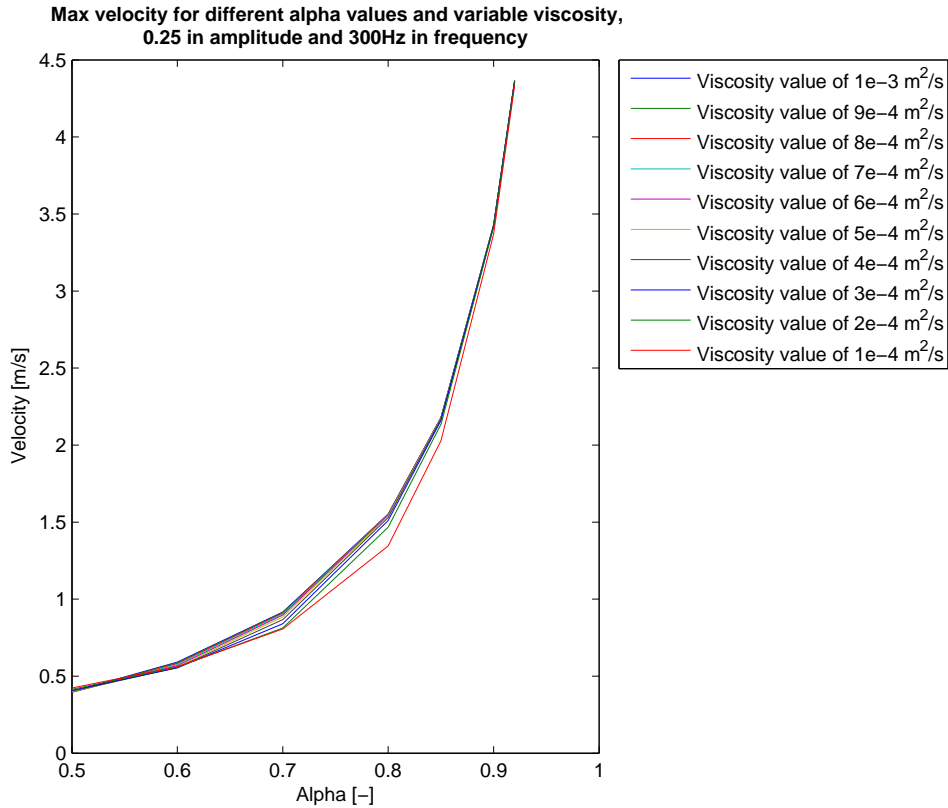


Figure 3.40: Max velocity as a function of alpha. Plotted for different viscosities.

All the different viscosities are following the same trendline and it can be seen that the viscosity do not affect the maximum velocity component much.

Chapter 4

Conclusion

4.1 Optimizing of simulation case

A common problem in CFD is long simulation times and in average each simulation done in this thesis has been of around 4 hours, and a reduction in the simulation time is very valuable when more than 400 simulations has been performed.

One way of reducing the simulation time is to utilize more processors during a simulation. This will reduce the simulation time, but this should first be done after the simulation case has been optimized in other ways. Some methods for optimising this case are explained here.

4.1.1 Adjustable time step

By using adjustable time step the simulation will adjust the time step accordingly so that it will always satisfy a set stability criteria that the user need to specify.

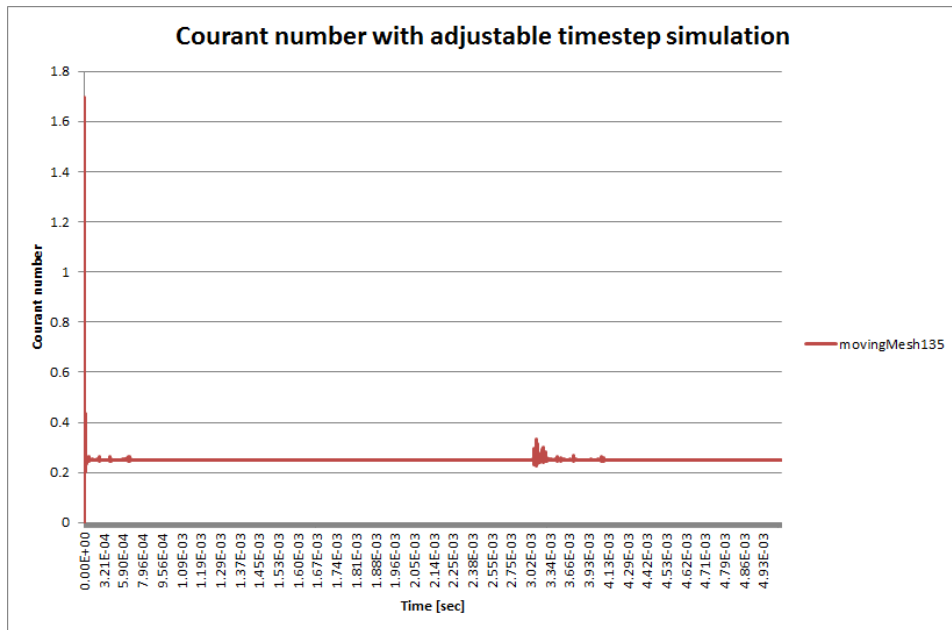


Figure 4.1: Courant number for adjustable time step simulation of the dynamic cylinder case.

The adjustable time-step algorithm is set such that it finds the largest possible time step below the set criteria and uses it as the simulation parameter. The criteria is set as a maximum allowable Courant number. To activate this you need to input *adjustTimeStep* yes; and *maxCo* max allowable Courant number; How this can be done is shown in attachment A.9 on line 48 to 50.

It can be seen in figure 4.1 on the preceding page that in the beginning of the simulation the Courant number criteria is not upheld and because of that the entire simulation is ruined. This happens even though the rest of the simulation is stable. A simulation will be ruined if only one time-step has a higher Courant number than the stability criteria. It will not converge due to the false premise that error in the stability creates.

4.1.2 Symmetry plane

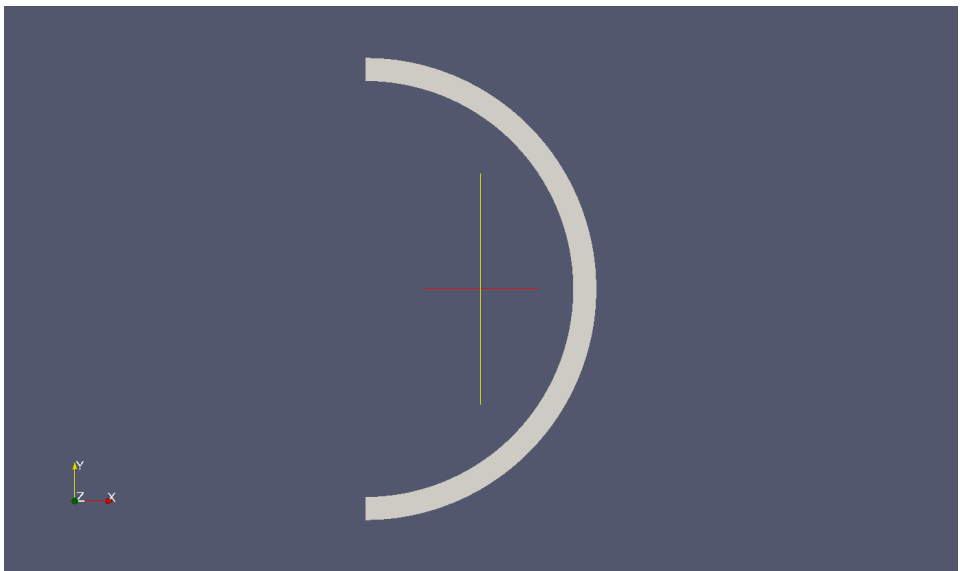


Figure 4.2: Figure of dynamic cylinder case with symmetry plane along the y-axis.

A great way of reducing the simulation time is to exploit the use of symmetry planes. For the dynamic cylinders case there is symmetry

along the y-axis. By implementing this symmetry plane, the simulation time can be reduced to half of the initial simulation time.

This method did not function properly for this simulation case, and it is most likely due to the sinusoidal acceleration input on the inner cylinder. The cells along the y-axis need to satisfy both the Neumann condition from the symmetry criteria as well as the moving boundary condition. This is not possible and this argument means that a symmetry plane is not available for this case.

4.1.3 Cell layer addition/removal

The diffusivity mesh method implies a strict timestep criteria in order to conserve stability over the entire simulation timespan. Courant numbers below 0.01 may be necessary to keep the simulation stable and that requires either a finer mesh or lower timesteps. Both of these changes increase the simulation time. This is required in order to keep the mesh skewness low during the dynamic movement of the mesh.

In order to avoid this strict time step criterion, an adaptive mesh method that adds layers of extra grid meshes based on how much the mesh is stretched may be used. This technique is called cell layering. It removes mesh layers if the distance between the meshes is too small and adds mesh if the distance is large. This technique removes mesh that increase the computational needs and adds mesh to keep the simulation stable. A method that is recommended to use for this type of case.

4.2 Practical experiment

During the timespan of the master thesis a practical experiment to verify the results found in the simulation data were planned. This has not been performed since there has been no specific findings to test in the simulated data.

One of the goals of the simulations has been to see if there is a transition phase in the span of the simulated data. This transition could have been seen in other aspects such as a drop in the forces on the inner cylinder, a laminar to turbulent transition or other clear flow changes. No such transition has been found and due to this the practical experiment was not performed.

4.3 Conclusion

4.3.1 Non dimensional study

The most interesting non-dimensional number found from this study was the B-value from the *Richardson's annular effect*. It is also directly linked to the kinetic Reynolds number. For this number a clear line between flow with and without near-wall velocity overshoot was found. This effect was seen in cases with a small alpha-value and high frequencies.

All other non-dimensional numbers that was compared was well within their stable range and there were no specific new findings.

4.3.2 Gradient model

The proposed gradient model did not prove to bring a new number for stability for this case and was discarded early in the thesis work.

4.3.3 Verification of model for the dynamic system

The dynamic model showed signs of trouble with stability for low frequency responses on the inner cylinder. In order to work around this problem, smaller time steps had to be used to keep the system stable, even though higher frequency responses have larger velocity gradients. The reason for this problem has not been found.

This master thesis is based on the preliminary work done in the project thesis and that may have biased the results, as it was expected to reproduce the results from that thesis with the new dynamic mesh model. All work done in this thesis is available for verification.

4.3.4 Further work

A small selection of viscosities and frequencies were selected for this master thesis due to the limited time span and further work on expanding the model would be preferable. Especially for larger Reynolds numbers to see how the transition from laminar to turbulent would be for this system. This thesis was restricted to laminar flow so no research on the turbulent effects of this system has been

performed. This restriction was set due to the need for different solvers for turbulent flow as well as the complexity of a turbulent study. There is not much known data on this type of system at the moment, and it was more realistic to perform a laminar study for this case first.

The model can easily be expanded to add inertia effects and more advanced input responses. Also interesting modelling options such as force optimisation may be coupled to the solver. There are many interesting effects that can be studied for this case in the future.

An amplitude study has been performed in this thesis, but large amplitude responses was not studied. They give small gap-size effects and this could be studied further.

Efforts to reduce the simulation time is important for CFD studies and this case can be improved in this area as well. Before a continued thorough study on this case, improvements in the simulation time should be achieved.

Some signs of high frequency effects has been noticed in the simulation data. These effects was beyond the scope of this thesis and has not been expanded further on. One specific effect that would be interesting to conduct further study on is the high frequency effects with low viscosity fluids. This can be performed as a individual study if possible.

4.3.5 Notes

A new version of OpenFOAM (2.20) with different file structure was released during this thesis work. The description for compiling the solver library is not the same as described for OpenFOAM 2.11. In order to compile the library the new file paths needs to be used.

Bibliography

- [1] J.H.Ferziger & Peric M. (2002) *Computational Methods for Fluid Dynamics*. third, rev edition, Berlin: Springer Verlag.
- [2] Veerstedeg H.K. & Malalasekera W. (2007) *Computational Fluid Dynamics*. second edition, Essex: Pearson Education Limited.
- [3] Shaw C. T. (1992) *Using Computational Fluid Dynamics*, England, Prentice Hall.
- [4] Edelsbrunner H. (2001) *Geometry and Topology for Mesh Generation*, England, Cambridge University Press.
- [5] Schlichting H. & Gersten K. (2005) *Boundary Layer Theory*, Berlin, Springer Verlag.
- [6] White, Frank M. (2009) *Fluid Mechanics, 7th Edition*, New York, McGraw-Hill Higher Education.
- [7] White, Frank M. (2005) *Viscous Fluid Flow, Third Edition*, Singapore, McGraw-Hill Education.
- [8] OpenFOAM *User Guide* [Online] OpenCFD Ltd. Available at: <http://foam.sourceforge.net/docs/Guides-a4/UserGuide.pdf> [Accessed 18 October 2012].
- [9] OpenFOAM *Programmers Guide* [Online] OpenCFD Ltd. Available at: <http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf> [Accessed 18 October 2012].
- [10] *Features of OpenFOAM*. [online] Available at: <http://www.openfoam.com/features/> [Accessed 19 September 2012].

- [11] *CFD Online Discussion Forums*. [online] Available at: <http://www.cfd-online.com/Forums/1> [Accessed 26 September 2012].
- [12] *ParaView - Open Source Scientific Visualization*. [online] Available at: <http://www.paraview.org/> [Accessed 26 November 2012].
- [13] Thompson J.E, Warsi Z.U.A. & Mastin W. (1997) *Numerical Grid Generation*, North-Holland
- [14] Øgård M. (2012) *Flow in cylindrical oil damper*, Norway, NTNU.
- [15] Nouar C. et al (1995) *Developing laminar flow in the entrance region of annuli - review and extension of standard resolution methods for the hydrodynamic problem*, France
- [16] Blom J. Fredric (1998) *Consideratons on the spring analogy*, International journal for numerical methods in fluids, 2000 (32), pp. 647-668
- [17] Jasak Hrvoje (2009) *Dynamic Mesh Handling in OpenFOAM*, American Institute of Aeronautics and Astronautics
- [18] Demirdzic I. & Peric M. (1988) *Space conservation law in finite volume calculations of fluid flow*, Int. J. Num. Meth. Fluids, 1988 (8), pp. 1037-1050
- [19] Jasak H. & Gosman A.D. (2000) *Automatic resolution control for the finite-volume method, part 2: adaptive mesh refinement and coarsening*, Numerical Heat Transfer, part B 2000 (38), pp. 257-271
- [20] Jasak Hrvoje & Tukovic Zeljko (2010) *Dynamic mesh handling in OpenFOAM applied to fluid-structure interaction simulations*, V European Conference of Computational Fluid Dynamics ECCOMAS CFD 2010, Lisboa, Portugal
- [21] Huang Weizhang & Russell Robert D. (2010) *Adaptive Moving Mesh Methods*, New York, Springer Science.
- [22] Jasak Hrvoje & Tukovic Zeljko (2004) *Automatic mesh motion for the unstructured finite volume method*, Transactions of FAMENA, v 30 n 2, 2007

- [23] Lucchini T. & D'Errico G. et al. (2007) *Automatic mesh motion with topological changes for engine simulation*, SAE Technical paper
- [24] Russell Robert D. & Huang Weizhang (1998) *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM Journal of Scientific Computing

Appendix A

Case files and Matlab scripts

A.1 U file for hydraulic piston case.

```
/*-----* C++ -*-----*\n|=====\n| \\ \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox\n| \\ \\ / O p e r a t i o n | Version: 2.1.1\n| \\ \\ / A n d | Web: www.OpenFOAM.org\n| \\ \\ / M a n i p u l a t i o n |\n/*-----*-----*\nFoamFile\n{\n  version      2.0;\n  format       ascii;\n  class        volVectorField;\n  object       U;\n}\n// ***** //\n\ndimensions      [0 1 -1 0 0 0];\n\ninternalField   uniform (0 0 0);\n\nboundaryField\n{\n  movingWall\n  {\n    type        movingWallVelocity;\n    value       uniform (0 0 0);\n  }\n\n  fixedWall\n  {\n    type        fixedValue;\n    value       uniform (0 0 0);\n  }\n\n  frontAndBack\n  {\n    type        empty;\n  }\n\n  axis\n  {\n    type        symmetryPlane;\n  }\n}\n\n// ***** //
```

A.2 p file for hydraulic piston case.

```
/*-----* C++ *-----*/
|=====|
|  \ \ /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration  | Version: 2.1.1
|  \ \ /  /  A nd        | Web:      www.OpenFOAM.org
|  \ \ /  /  M anipulation|
/*-----*

FoamFile
{
    version      2.0;
    format       ascii;
    class        volScalarField;
    object       p;
}
// *****

dimensions      [0 2 -2 0 0 0];

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type      zeroGradient;
    }

    fixedWall
    {
        type      zeroGradient;
    }

    frontAndBack
    {
        type      empty;
    }

    axis
    {
        type      symmetryPlane;
    }
}

// *****
```

A.3 pointMotionU file for hydraulic piston case.

```

/*-----* C++ -*-----*/
|=====|
|  \ \ /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration  | Version: 2.1.1
|  \ \ /  /  A nd        | Web:      www.OpenFOAM.org
|  \ \ /  /  M anipulation|
/*-----*

```

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        pointVectorField;
    object       pointMotionU;
}
// *****

```

```

dimensions      [0 1 -1 0 0 0];

```

```

internalField   uniform (0 0 0);

```

```

boundaryField
{
    movingWall
    {
        type timeVaryingUniformFixedValue;
        fileName "$FOAM_CASE/0/hyd_cylinder19.dat";
        outOfBounds warn;
    }

    fixedWall
    {
        type          fixedValue;
        value         uniform (0 0 0);
    }

    frontAndBack
    {
        type          empty;
    }

    axis
    {
        type          symmetryPlane;
    }
}

```

```

// *****

```

A.4 blockMeshDict file for hydraulic piston case.

```

/*-----* C++ *-----*/
|=====|
| \ / | F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \ / | O p e r a t i o n | Version: 2.1.0
| \ / | A n d | Web: www.OpenFOAM.org
| \ / | M a n i p u l a t i o n |
/*-----*

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       blockMeshDict;
}
// *****

convertToMeters 0.001;

vertices
(
    (0 0 0) // 0
    (2 0 0) // 1
    (2 2.0 0) // 2
    (5 2.0 0) // 3
    (5 0 0) // 4
    (8 0 0) // 5
    (8 2.0 0) // 6
    (8 2.25 0) // 7
    (5 2.25 0) // 8
    (2 2.25 0) // 9
    (0 2.25 0) // 10
    (0 2.0 0) // 11

    (0 0 1) // 12
    (2 0 1) // 13
    (2 2.0 1) // 14
    (5 2.0 1) // 15
    (5 0 1) // 16
    (8 0 1) // 17
    (8 2.0 1) // 18
    (8 2.25 1) // 19
    (5 2.25 1) // 20
    (2 2.25 1) // 21
    (0 2.25 1) // 22
    (0 2.0 1) // 23

);
blocks
(
    hex (0 1 2 11 12 13 14 23) (75 100 1) simpleGrading (1 1 1) // lower left corner
    hex (11 2 9 10 23 14 21 22) (75 50 1) simpleGrading (1 1 1) // upper left corner
    hex (2 3 8 9 14 15 20 21) (175 50 1) simpleGrading (1 1 1) // channel
    hex (4 5 6 3 16 17 18 15) (150 150 1) simpleGrading (1 1 1) // right lower corner
    hex (3 6 7 8 15 18 19 20) (150 50 1) simpleGrading (1 1 1) // right upper corner
);
edges
(
);
boundary
(
    frontAndBack
    {

```



```
type empty;
faces
(
    (0 1 2 11)
    (11 2 9 10)
    (2 3 8 9)
    (4 5 6 3)
    (3 6 7 8)

    (12 13 14 23)
    (23 14 21 22)
    (14 15 20 21)
    (16 17 18 15)
    (15 18 19 20)
);
}

movingWall
{
    type wall;
    faces
    (
        (1 2 14 13)
        (3 15 14 2)
        (4 16 15 3)
    );
}

fixedWall
{
    type wall;
    faces
    (
        (9 21 22 10)
        (8 20 21 9)
        (7 19 20 8)

        (0 11 23 12)
        (11 10 22 23)

        (5 17 18 6)
        (6 18 19 7)
    );
}

axis
{
    type symmetryPlane;
    faces
    (
        (0 1 13 12)
        (5 17 16 4)
    );
}

};

mergePatchPairs
(
);

// ***** //
```

A.5 U file for dynamic simulation of the concentric cylinder case.

```
/*-----* C++ -*-----*\
|=====|
|  \ \ /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration  | Version: 2.1.1
|  \ \ /  /  A nd        | Web:      www.OpenFOAM.org
|  \ \ /  /  M anipulation|
|-----*\
FoamFile
{
    version      2.0;
    format       ascii;
    class        volVectorField;
    object       U;
}
// *****

dimensions      [0 1 -1 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type      movingWallVelocity;
        value      uniform (0 0 0);
    }

    fixedWall
    {
        type      fixedValue;
        value      uniform (0 0 0);
    }

    frontAndBack
    {
        type      empty;
    }

    axis
    {
        type      symmetryPlane;
    }
}

// *****
```

A.6 p file for dynamic simulation of the concentric cylinder case.

```
/*-----* C++ *-----*\
|=====|
|  \ \ /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration  | Version: 2.1.1
|  \ \ /  /  A nd         | Web:      www.OpenFOAM.org
|  \ \ /  /  M anipulation|
\*-----*\
FoamFile
{
    version      2.0;
    format       ascii;
    class        volScalarField;
    object       p;
}
// *****

dimensions      [0 2 -2 0 0 0];

internalField   uniform 0;

boundaryField
{
    movingWall
    {
        type      zeroGradient;
    }

    fixedWall
    {
        type      zeroGradient;
    }

    frontAndBack
    {
        type      empty;
    }

    axis
    {
        type      symmetryPlane;
    }
}

// *****
```

A.7 pointDisplacement file for dynamic simulation of the concentric cylinder case.

```

/*-----* C++ -*-----*/
|=====|
|  \ \ /  /  F ield      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration  | Version: 2.1.1
|  \ \ /  /  A nd        | Web:      www.OpenFOAM.org
|  \ \ /  /  M anipulation|
/*-----*

FoamFile
{
    version      2.0;
    format       ascii;
    class        pointVectorField;
    object       pointDisplacement;
}
// *****

dimensions      [0 1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    movingWall
    {
        type oscillatingDisplacement;
        amplitude (0 0.00025 0);
        omega 314.16; // 50 hertz
        angle0 0;
        axis (0 0 1);
        origin (0 0 0);
        value uniform (0 0 0);
    }

    fixedWall
    {
        type          fixedValue;
        value         uniform (0 0 0);
    }

    frontAndBack
    {
        type          empty;
    }
}

// *****

```

A.8 blockMeshDict file for dynamic simulation of the concentric cylinder case.


```

/*-----* C++ *-----*/
|=====|
| \ \ / / F ield | OpenFOAM: The Open Source CFD Toolbox
| \ \ / / O peration | Version: 2.1.0
| \ \ / / A nd | Web: www.OpenFOAM.org
| \ \ / / M anipulation |
/*-----*/

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       blockMeshDict;
}
// *****

convertToMeters 0.010;

vertices
(
// Front circle
(0.900 0.000 0.0) // 0
(0.636396 0.636396 0.0) // 1
(0.0 0.900 0.0) // 2
(-0.636396 0.636396 0.0) // 3
(-0.900 0.000 0.0) // 4
(-0.636396 -0.636396 0.0) // 5
(0.0 -0.900 0.0) // 6
(0.636396 -0.636396 0.0) // 7

(1.000 0.0 0.0) // 8
(0.707107 0.707107 0.0) // 9
(0.0 1.000 0.0) // 10
(-0.707107 0.707107 0.0) // 11
(-1.000 0.0 0.0) // 12
(-0.707107 -0.707107 0.0) // 13
(0.0 -1.000 0.0) // 14
(0.707107 -0.707107 0.0) // 15
// Rear circle
(0.900 0.000 1.000) // 16
(0.636396 0.636396 1.000) // 17
(0.0 0.900 1.000) // 18
(-0.636396 0.636396 1.000) // 19
(-0.900 0.000 1.000) // 20
(-0.636396 -0.636396 1.000) // 21
(0.0 -0.900 1.000) // 22
(0.636396 -0.636396 1.000) // 23

(1.000 0.0 1.000) // 24
(0.707107 0.707107 1.000) // 25
(0.0 1.000 1.000) // 26
(-0.707107 0.707107 1.000) // 27
(-1.000 0.0 1.000) // 28
(-0.707107 -0.707107 1.000) // 29
(0.0 -1.000 1.000) // 30
(0.707107 -0.707107 1.000) // 31
);
blocks
(
hex (0 8 9 1 16 24 25 17) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (2 1 9 10 18 17 25 26) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (3 2 10 11 19 18 26 27) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (12 4 3 11 28 20 19 27) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (13 5 4 12 29 21 20 28) (40 40 1) simpleGrading (1.0 1.0 1.0)
)

```

```
hex (13 14 6 5 29 30 22 21) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (14 15 7 6 30 31 23 22) (40 40 1) simpleGrading (1.0 1.0 1.0)
hex (7 15 8 0 23 31 24 16) (40 40 1) simpleGrading (1.0 1.0 1.0)
);
```

edges

```
(
arc 0 1 (0.831492 0.344415 0.0)
arc 1 2 (0.344415 0.831492 0.0)
arc 2 3 (-0.344415 0.831492 0.0)
arc 3 4 (-0.831492 0.344415 0.0)
arc 4 5 (-0.831492 -0.344415 0.0)
arc 5 6 (-0.344415 -0.831492 0.0)
arc 6 7 (0.344415 -0.831492 0.0)
arc 7 0 (0.831492 -0.344415 0.0)
arc 8 9 (0.923880 0.382683 0.0)
arc 9 10 (0.382683 0.923880 0.0)
arc 10 11 (-0.382683 0.923880 0.0)
arc 11 12 (-0.923880 0.382683 0.0)
arc 12 13 (-0.923880 -0.382683 0.0)
arc 13 14 (-0.382683 -0.923880 0.0)
arc 14 15 (0.382683 -0.923880 0.0)
arc 15 8 (0.923880 -0.382683 0.0)
arc 16 17 (0.831492 0.344415 1.0)
arc 17 18 (0.344415 0.831492 1.0)
arc 18 19 (-0.344415 0.831492 1.0)
arc 19 20 (-0.831492 0.344415 1.0)
arc 20 21 (-0.831492 -0.344415 1.0)
arc 21 22 (-0.344415 -0.831492 1.0)
arc 22 23 (0.344415 -0.831492 1.0)
arc 23 16 (0.831492 -0.344415 1.0)
arc 24 25 (0.923880 0.382683 1.0)
arc 25 26 (0.382683 0.923880 1.0)
arc 26 27 (-0.382683 0.923880 1.0)
arc 27 28 (-0.923880 0.382683 1.0)
arc 28 29 (-0.923880 -0.382683 1.0)
arc 29 30 (-0.382683 -0.923880 1.0)
arc 30 31 (0.382683 -0.923880 1.0)
arc 31 24 (0.923880 -0.382683 1.0)
);
```

boundary

```
(
frontAndBack
{
type empty;
faces
(
(0 8 9 1)
(1 9 10 2)
(2 10 11 3)
(3 11 12 4)
(4 12 13 5)
(5 13 14 6)
(6 14 15 7)
(7 15 8 0)
(16 24 25 17)
(17 25 26 18)
(18 26 27 19)
(19 27 28 20)
(20 28 29 21)
(21 29 30 22)
(22 30 31 23)
(23 31 24 16)

```

```
);
}
movingWall
{
type wall;
faces
(
(0 1 17 16)
(1 2 18 17)
(2 3 19 18)
(3 4 20 19)
(4 5 21 20)
(5 6 22 21)
(6 7 23 22)
(7 0 16 23)
);
}
fixedWall
{
type wall;
faces
(
(9 8 24 25)
(10 9 25 26)
(11 10 26 27)
(12 11 27 28)
(13 12 28 29)
(14 13 29 30)
(15 14 30 31)
(8 15 31 24)
);
}
}
);

mergePatchPairs
(
);

// ***** //
```

A.9 controlDict file for dynamic simulation of the concentric cylinder case.

```

1  /*-----* C++ *-----*/
2  |=====|
3  |  \  /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
4  |  \  /  O peration    | Version: 2.1.1
5  |  \  /  A n d         | Web: www.OpenFOAM.org
6  |  \  /  M anipulation  |
7  /*-----*-----*/
8  FoamFile
9  {
10     version      2.0;
11     format       ascii;
12     class        dictionary;
13     location     "system";
14     object       controlDict;
15 }
16 // *****
17
18 application     pimpleDyMFoam;
19
20 startFrom       startTime;
21
22 startTime       0;
23
24 stopAt          endTime;
25
26 endTime        0.005;
27
28 deltaT         1e-08;
29
30 writeControl    timeStep;
31
32 writeInterval   2000;
33
34 purgeWrite     0;
35
36 writeFormat     binary;
37
38 writePrecision  6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 runtimeModifiable true;
47
48 adjustTimeStep no;
49
50 maxCo          0.2;
51
52 libs ("libMyFunctionDisplacement.so");
53
54
55 functions
56 {
57     #include "forces"
58 }
59
60 // *****
61

```

A.10 forces file for dynamic simulation of the concentric cylinder case.

```
1  forces
2  {
3  type forces;
4  functionObjectLibs ("libforces.so");
5  patches ("movingWall.*");
6  pName p;
7  UName U;
8  rhoName rhoInf;
9  log true;
10 rhoInf 800.0;
11 CofR (0 0 0);
12 outputControl timeStep;
13 outputInterval 1;
14 }
15 );
16
```

A.11 angularOscillatingDisplacement PointPatchVectorField.C file.


```

1  /*-----*\
2  =====
3  \ \ / / F ield      | OpenFOAM: The Open Source CFD Toolbox
4  \ \ / / O peration  |
5  \ \ / / A nd         | Copyright (C) 2011 OpenFOAM Foundation
6  \ \ / / M anipulation |
7  -----*\
8  License
9  This file is part of OpenFOAM.
10
11  OpenFOAM is free software: you can redistribute it and/or modify it
12  under the terms of the GNU General Public License as published by
13  the Free Software Foundation, either version 3 of the License, or
14  (at your option) any later version.
15
16  OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18  FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
19  for more details.
20
21  You should have received a copy of the GNU General Public License
22  along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.
23
24  \*-----*/
25
26  #include "angularOscillatingDisplacementPointPatchVectorField.H"
27  #include "pointPatchFields.H"
28  #include "addToRunTimeSelectionTable.H"
29  #include "Time.H"
30  #include "polyMesh.H"
31
32  // * * * * * //
33
34  namespace Foam
35  {
36
37  // * * * * * Constructors * * * * * //
38
39  angularOscillatingDisplacementPointPatchVectorField::
40  angularOscillatingDisplacementPointPatchVectorField
41  (
42      const pointPatch& p,
43      const DimensionedField<vector, pointMesh>& iF
44  )
45  :
46      fixedValuePointPatchField<vector>(p, iF),
47      axis_(vector::zero),
48      origin_(vector::zero),
49      angle0_(0.0),
50      amplitude_(0.0),
51      omega_(0.0),
52      p0_(p.localPoints())
53  {}
54
55
56  angularOscillatingDisplacementPointPatchVectorField::
57  angularOscillatingDisplacementPointPatchVectorField
58  (
59      const pointPatch& p,
60      const DimensionedField<vector, pointMesh>& iF,
61      const dictionary& dict
62  )
63  :
64      fixedValuePointPatchField<vector>(p, iF, dict),

```

```

65     axis_(dict.lookup("axis")),
66     origin_(dict.lookup("origin")),
67     angle0_(readScalar(dict.lookup("angle0"))),
68     amplitude_(readScalar(dict.lookup("amplitude"))),
69     omega_(readScalar(dict.lookup("omega")))
70 {
71     if (!dict.found("value"))
72     {
73         updateCoeffs();
74     }
75
76     if (dict.found("p0"))
77     {
78         p0_ = vectorField("p0", dict , p.size());
79     }
80     else
81     {
82         p0_ = p.localPoints();
83     }
84 }
85
86
87 angularOscillatingDisplacementPointPatchVectorField::
88 angularOscillatingDisplacementPointPatchVectorField
89 (
90     const angularOscillatingDisplacementPointPatchVectorField& ptf,
91     const pointPatch& p,
92     const DimensionedField<vector, pointMesh>& iF,
93     const pointPatchFieldMapper& mapper
94 )
95 :
96     fixedValuePointPatchField<vector>(ptf, p, iF, mapper),
97     axis_(ptf.axis_),
98     origin_(ptf.origin_),
99     angle0_(ptf.angle0_),
100    amplitude_(ptf.amplitude_),
101    omega_(ptf.omega_),
102    p0_(ptf.p0_, mapper)
103 {}
104
105
106 angularOscillatingDisplacementPointPatchVectorField::
107 angularOscillatingDisplacementPointPatchVectorField
108 (
109     const angularOscillatingDisplacementPointPatchVectorField& ptf,
110     const DimensionedField<vector, pointMesh>& iF
111 )
112 :
113     fixedValuePointPatchField<vector>(ptf, iF),
114     axis_(ptf.axis_),
115     origin_(ptf.origin_),
116     angle0_(ptf.angle0_),
117     amplitude_(ptf.amplitude_),
118     omega_(ptf.omega_),
119     p0_(ptf.p0_)
120 {}
121
122
123 // * * * * * Member Functions * * * * * //
124
125 void angularOscillatingDisplacementPointPatchVectorField::autoMap
126 (
127     const pointPatchFieldMapper& m
128 )

```

```

129 {
130     fixedValuePointPatchField<vector>::autoMap(m);
131
132     p0_.autoMap(m);
133 }
134
135
136 void angularOscillatingDisplacementPointPatchVectorField::rmap
137 (
138     const pointPatchField<vector>& ptf,
139     const labelList& addr
140 )
141 {
142     const angularOscillatingDisplacementPointPatchVectorField& aODptf =
143         refCast<const angularOscillatingDisplacementPointPatchVectorField>(ptf);
144
145     fixedValuePointPatchField<vector>::rmap(aODptf, addr);
146
147     p0_.rmap(aODptf.p0_, addr);
148 }
149
150
151 void angularOscillatingDisplacementPointPatchVectorField::updateCoeffs()
152 {
153     if (this->updated())
154     {
155         return;
156     }
157
158     const polyMesh& mesh = this->dimensionedInternalField().mesh();
159     const Time& t = mesh.time();
160
161     scalar angle = angle0_ + amplitude_*sin(omega_*t.value()); // Here you can see the
formula
162     vector axisHat = axis_/mag(axis_);
163     vectorField p0Rel(p0_ - origin_);
164
165     vectorField::operator=
166     (
167         p0Rel*(cos(angle) - 1)
168         + (axisHat ^ p0Rel*sin(angle))
169         + (axisHat & p0Rel)*(1 - cos(angle))*axisHat
170     );
171
172     fixedValuePointPatchField<vector>::updateCoeffs();
173 }
174
175
176 void angularOscillatingDisplacementPointPatchVectorField::write
177 (
178     Ostream& os
179 ) const
180 {
181     pointPatchField<vector>::write(os);
182     os.writeKeyword("axis")
183         << axis_ << token::END_STATEMENT << nl;
184     os.writeKeyword("origin")
185         << origin_ << token::END_STATEMENT << nl;
186     os.writeKeyword("angle0")
187         << angle0_ << token::END_STATEMENT << nl;
188     os.writeKeyword("amplitude")
189         << amplitude_ << token::END_STATEMENT << nl;
190     os.writeKeyword("omega")
191         << omega_ << token::END_STATEMENT << nl;

```


A.12 `oscillatingDisplacementPointPatch VectorField.C`
file.

```

1  /*-----*\
2  =====
3  \ \ / / F ield      | OpenFOAM: The Open Source CFD Toolbox
4  \ \ / / O peration  |
5  \ \ / / A nd        | Copyright (C) 2011 OpenFOAM Foundation
6  \ \ / / M anipulation |
7  -----*\
8  License
9      This file is part of OpenFOAM.
10
11     OpenFOAM is free software: you can redistribute it and/or modify it
12     under the terms of the GNU General Public License as published by
13     the Free Software Foundation, either version 3 of the License, or
14     (at your option) any later version.
15
16     OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17     ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18     FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
19     for more details.
20
21     You should have received a copy of the GNU General Public License
22     along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.
23
24  \*-----*/
25
26  #include "oscillatingDisplacementPointPatchVectorField.H"
27  #include "pointPatchFields.H"
28  #include "addToRunTimeSelectionTable.H"
29  #include "Time.H"
30  #include "polyMesh.H"
31
32  // * * * * *
33
34  namespace Foam
35  {
36
37  // * * * * * Constructors * * * * *
38
39  oscillatingDisplacementPointPatchVectorField::
40  oscillatingDisplacementPointPatchVectorField
41  (
42      const pointPatch& p,
43      const DimensionedField<vector, pointMesh>& iF
44  )
45  :
46      fixedValuePointPatchField<vector>(p, iF),
47      amplitude_(vector::zero),
48      omega_(0.0)
49  {}
50
51
52  oscillatingDisplacementPointPatchVectorField::
53  oscillatingDisplacementPointPatchVectorField
54  (
55      const pointPatch& p,
56      const DimensionedField<vector, pointMesh>& iF,
57      const dictionary& dict
58  )
59  :
60      fixedValuePointPatchField<vector>(p, iF, dict),
61      amplitude_(dict.lookup("amplitude")),
62      omega_(readScalar(dict.lookup("omega")))
63  {
64      if (!dict.found("value"))

```


A.13 `libMyFunctionDisplacementPointPatch` `VectorField.C` file.

```

1  /*-----*\
2  =====
3  \ \ / / F ield      | OpenFOAM: The Open Source CFD Toolbox
4  \ \ / / O peration  |
5  \ \ / / A nd         | Copyright (C) 2011 OpenFOAM Foundation
6  \ \ / / M anipulation |
7  -----*\
8  License
9  This file is part of OpenFOAM.
10
11  OpenFOAM is free software: you can redistribute it and/or modify it
12  under the terms of the GNU General Public License as published by
13  the Free Software Foundation, either version 3 of the License, or
14  (at your option) any later version.
15
16  OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
17  ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
18  FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
19  for more details.
20
21  You should have received a copy of the GNU General Public License
22  along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.
23
24  \*-----*/
25
26  #include "libMyFunctionPointPatchVectorField.H"
27  #include "pointPatchFields.H"
28  #include "addToRunTimeSelectionTable.H"
29  #include "Time.H"
30  #include "polyMesh.H"
31
32  // * * * * * //
33
34  namespace Foam
35  {
36
37  // * * * * * Constructors * * * * * //
38
39  libMyFunctionPointPatchVectorField::
40  libMyFunctionPointPatchVectorField
41  (
42      const pointPatch& p,
43      const DimensionedField<vector, pointMesh>& iF
44  )
45  :
46      fixedValuePointPatchField<vector>(p, iF),
47      amplitude_(vector::zero),
48      omega_(0.0)
49  {}
50
51
52  libMyFunctionPointPatchVectorField::
53  libMyFunctionPointPatchVectorField
54  (
55      const pointPatch& p,
56      const DimensionedField<vector, pointMesh>& iF,
57      const dictionary& dict
58  )
59  :
60      fixedValuePointPatchField<vector>(p, iF, dict),
61      amplitude_(dict.lookup("amplitude")),
62      omega_(readScalar(dict.lookup("omega")))
63  {
64      if (!dict.found("value"))

```

```
65     {
66         updateCoeffs();
67     }
68 }
69
70
71 libMyFunctionPointPatchVectorField::
72 libMyFunctionPointPatchVectorField
73 (
74     const libMyFunctionPointPatchVectorField& ptf,
75     pointPatch& p,
76     const DimensionedField<vector, pointMesh>& iF,
77     const pointPatchFieldMapper& mapper
78 )
79 :
80     fixedValuePointPatchField<vector>(ptf, p, iF, mapper),
81     amplitude_(ptf.amplitude_),
82     omega_(ptf.omega_)
83 {}
84
85
86 libMyFunctionPointPatchVectorField::
87 libMyFunctionPointPatchVectorField
88 (
89     const libMyFunctionPointPatchVectorField& ptf,
90     const DimensionedField<vector, pointMesh>& iF
91 )
92 :
93     fixedValuePointPatchField<vector>(ptf, iF),
94     amplitude_(ptf.amplitude_),
95     omega_(ptf.omega_)
96 {}
97
98
99 // * * * * * Member Functions * * * * * //
100
101 void libMyFunctionPointPatchVectorField::updateCoeffs()
102 {
103     if (this->updated())
104     {
105         return;
106     }
107
108     const polyMesh& mesh = this->dimensionedInternalField().mesh();
109     const Time& t = mesh.time();
110
111     // Field<vector>::operator=(amplitude_*sin(omega_*t.value())); // initial function
112
113     Field<vector>::operator=((amplitude_/omega_)*(t.value()-sin(omega_*t.value()))); //
114     new function by Marius Øgård
115
116     fixedValuePointPatchField<vector>::updateCoeffs();
117 }
118
119 void libMyFunctionPointPatchVectorField::write(Ostream& os) const
120 {
121     pointPatchField<vector>::write(os);
122     os.writeKeyword("amplitude")
123         << amplitude_ << token::END_STATEMENT << nl;
124     os.writeKeyword("omega")
125         << omega_ << token::END_STATEMENT << nl;
126     writeEntry("value", os);
127 }
```

```
128
129
130 // * * * * * //
131
132 makePointPatchTypeField
133 (
134     pointPatchVectorField,
135     libMyFunctionPointPatchVectorField
136 );
137
138 // * * * * * //
139
140 } // End namespace Foam
141
142 // ***** //
143
```

A.14 pointDisplacement file for use with modified library.

```

1  /*-----* C++ *-----*/
2  |=====|
3  |  \ \ /  F ield      | OpenFOAM: The Open Source CFD Toolbox
4  |  \ \ /  O peration  | Version: 2.1.1
5  |  \ \ /  A nd        | Web: www.OpenFOAM.org
6  |  \ \ /  M anipulation |
7  /*-----*
8  FoamFile
9  {
10     version      2.0;
11     format        ascii;
12     class         pointVectorField;
13     object        pointDisplacement;
14 }
15 // *****
16
17 dimensions      [0 1 0 0 0 0];
18
19 internalField   uniform (0 0 0);
20
21 boundaryField
22 {
23     movingWall
24     {
25     type libMyFunction;
26     amplitude (0 0.25 0);
27     omega 3141.59;
28     value uniform (0 0 0);
29     }
30
31     fixedWall
32     {
33     type          fixedValue;
34     value         uniform (0 0 0);
35     }
36
37     frontAndBack
38     {
39     type          empty;
40     }
41 }
42 }
43
44 // *****
45

```

A.15 Matlab script for generating .dat file for use in pointMotionU file.

```
% Copyright Marius Øgård
% Script for generating .dat file for use in timeVaryingUniformFixedValue
% with velocityLaplacian as solver in dynamicMeshDict file.
% Only velocity in y-direction in this file

t_max = 0.005; % [seconds]

dt = 1e-7;

N = t_max/dt;

t = [0:dt:t_max];

A = 0.1; % Amplitude factor for velocity component [m/s]
phi = 0; % Phase shifting constant
freq = 250; % Frequency [Hz]
omega = 2*pi*freq; %

y = A.*sin(omega.*t); % Velocity function for y = A*sin(omega*t)

x = zeros(1,length(y));
z = zeros(1,length(y));

fileID = fopen('sin_A0.1_freq250_dt1e-7.dat','w+');

fprintf(fileID, ' ( \r\n');

% fprintf(fileID, ' ( 0 ( 0 0 0 )) \r\n')

for i=1:N

    fprintf(fileID, ' ( %1.7f ( %1.6f %1.7f %1.6f ) ) \r\n', (dt*i),x(i),y(i),z(i));

end

fprintf(fileID, ' ) \r\n');

fclose(fileID);
```


A.16 Matlab script for plotting of analytical solution for pressure gradient.

```

1  clear all
2  clc
3
4  delta_D = [0.5 0.6 0.7 0.8 0.9 0.92 0.94 0.96 0.98];
5
6  alpha = 0.9;
7  r2 = 10e-3;
8  r1 = alpha*r2;
9
10 dt = 1e-7;
11 t_max = 0.005;
12
13 t = [0:dt:t_max];
14
15 rho = 1000;
16 ny = 0.001;
17 mu = ny*rho;
18 freq = 300; % [Hertz]
19 omega = 2*pi*freq;
20 A = 0.25;
21
22 a = A.*sin(omega.*t);
23 v = (A./omega).*(1-cos(omega.*t));
24 y = (A./omega).*(t-(sin(omega.*t)./omega));
25
26 % A(1) = (ri^2*ln(ri)-r2^2*ln(r2))/(2*(r2^2-ri^2))*;
27 % B(1) = (ri^2*r2^2)/2*(r2^2-ri^2)*ln();
28
29 K_alpha(1) = (r2/(4*(gamma(1)-1)))*(((2*gamma(1)*log(gamma(1))^2)/(gamma(1)^2-1))-(gamma
(1)^2-1)/(2*gamma(1)));
30 K_beta(1) = (1/2*(gamma(1)-1))*((2*gamma(1)^2*log(gamma(1)))/(gamma(1)^2-1)-1);
31
32 N=length(t);
33
34 dtheta = 1; %
35 theta = [1:dtheta:360].*((2*pi)/360);
36 N_theta = length(theta);
37
38 gamma = zeros(N,N_theta);
39 ri = zeros(N,N_theta);
40 dP_dtheta = zeros(N,N_theta);
41
42 ri(1,:) = r1;
43 gamma(1,:) = r2/r1;
44
45     for i=2:N
46
47         ri(i,:) = sqrt(r1.^2+(y(i)-y(i-1))^2+2.*r1.*(y(i)-y(i-1)).*sin(theta(:)));
48
49         gamma(i,:) = r2./ri(i,:);
50
51         %         A(i) = ;
52         %         B(i) = ;
53
54         K_alpha(i) = (r2/(4*(gamma(i)-1)))*(((2*gamma(i)*log(gamma(i))^2)/(gamma(i)^2-1
))-(gamma(i)^2-1)/(2*gamma(i)));
55
56         K_beta(i) = (1/2*(gamma(i)-1))*((2*gamma(i)^2*log(gamma(i)))/(gamma(i)^2-1)-1);
57
58         dP_dtheta(i,:) = -(mu./K_alpha(i)).*((r1./ri(i,:))/(gamma(i,:)-1)+K_beta(i)).*v(
i).*cos(theta(:));
59
60     end
61

```

```
62
63     d_angle = 10;
64     angle_max = 90;
65     N_angle = 1+angle_max/d_angle;
66     angle = [0:d_angle:angle_max];
67
68     for j=1:N_angle
69
70         leg(j,:) = {sprintf('Theta angle of %2.0f',(j-1)*d_angle)};
71
72         x = t_max*(0:(1/(length(dP_dtheta)-1)):1);
73         fig1=figure(1);
74         hold all
75     %     xlim([0 360])
76
77     if (j-1) == 0
78
79         plot(x,dP_dtheta(:,360))
80         grad_max(1) = max(dP_dtheta(:,360));
81
82     else
83
84         plot(x,dP_dtheta(:,(j-1)*d_angle))
85         grad_max(j) = max(dP_dtheta(:,(j-1)*d_angle));
86
87     end
88
89     end
90
91     xlabel('Time [sec]')
92     ylabel('Pressure gradient [N/mm]')
93     legend(leg,'Location','NorthEastOutside')
94     title_handle = {'Analytical solution for pressure gradient, Alpha = 0.9, 1e-3 m^2/s in
95     viscosity,';...
96     '0.25 in amplitude and frequency of 300 Hz.'};
97     title(title_handle,'FontWeight','bold')
98     set(gca, 'ActivePositionProperty', 'OuterPosition');
99     fig_name = 'pressure_gradient';
100    print(fig1,'-djpeg',fig_name)
101
102
103    fig2=figure(2);
104    plot(angle,grad_max)
105    xlim([0 90])
106    xlabel('Angle [degrees]')
107    ylabel('Pressure gradient [N/mm]')
108    legend('Max pressure gradient','Location','NorthEastOutside')
109    title_handle = {'Analytical solution for pressure gradient, Alpha = 0.9, 1e-3 m^2/s in
110    viscosity,';...
111    '0.25 in amplitude and frequency of 300 Hz.'};
112    title(title_handle,'FontWeight','bold')
113    set(gca, 'ActivePositionProperty', 'OuterPosition');
114    fig_name = 'max_pressure_gradient';
115    print(fig2,'-djpeg',fig_name)
116
```

A.17 Matlab script for autogeneration of simulated velocity profiles.

```
1  clc
2  clear all
3
4  teller = 1;
5  k_teller = 1;
6
7  alpha = {sprintf('05'),sprintf('06'),sprintf('07'),sprintf('08'),sprintf('085'),sprintf(
8  '09'),sprintf('092')};
9  alpha_num = [0.5 0.6 0.7 0.8 0.85 0.9 0.92];
10 N_alpha = length(alpha);
11
12 visc = {sprintf('1e-3'),sprintf('9e-4'),sprintf('8e-4'),sprintf('7e-4'),sprintf('6e-4'),
13 sprintf('5e-4'),sprintf('4e-4'),sprintf('3e-4'),sprintf('2e-4'),sprintf('1e-4')};
14 visc_num = [1e-3 9e-4 8e-4 7e-4 6e-4 5e-4 4e-4 3e-4 2e-4 1e-4];
15 N_visc = length(visc);
16
17 start_time = 0.00000;
18 end_time = 0.00500;
19 dt = 2e-5;
20 time = start_time:dt:end_time;
21 N_time = length(time);
22
23 stored_max = 0;
24 temp_max = 0;
25
26 for iter=1:N_alpha
27     for jter=1:N_visc
28         for kter=1:N_time
29
30             k_teller = k_teller + 1;
31
32             temp = sprintf('visc%s_alpha%s/sets/%1.5f/zero_deg_U.xy',visc{jter},alpha{iter},
33 time(kter));
34
35             A = importdata(temp);
36
37             a = A(:,3);
38
39             temp_max = max(a);
40             mean_vel(kter) = mean(a);
41
42             if temp_max > stored_max
43                 stored_max = temp_max;
44
45             else
46                 temp_max = 0;
47
48             end
49
50         end
51
52     end
53
54     mean_vel = mean(mean_vel);
55
56     max_vel(jter,iter) = stored_max;
57     mean_vel_stored(jter,iter) = mean_vel;
58
59     stored_max = 0;
60
61
```

```
62     end
63
64     leg(iter,1) = {sprintf('Alpha value of %s',alpha{iter})};
65     leg2(iter,1) = {sprintf('Viscosity value of %s m^2/s',visc{jter})};
66
67     hold all
68
69     figure(teller)
70     plot(visc_num, max_vel(:, iter))
71
72     xlabel('Viscosity [m^2/s]')
73     ylabel('Velocity [m/s]')
74     legend('Max velocity', 'Location', 'NorthEastOutside')
75     title_handle = {sprintf('Max velocity for alpha value of %s and variable
viscosity, 'alpha{iter}); '0.25 in amplitude and 300Hz in frequency'};
76     title(title_handle, 'FontWeight', 'bold')
77     set(gca, 'ActivePositionProperty', 'OuterPosition');
78     fig_name = sprintf('max_velocity_visc%s_alpha%s', visc{jter}, alpha{iter});
79     print(figure(teller), '-djpeg', fig_name)
80
81     hold off
82
83     teller = teller + 1;
84
85 end
86
87
88     figure(teller)
89     plot(visc_num, max_vel)
90
91     xlabel('Viscosity [m^2/s]')
92     ylabel('Velocity [m/s]')
93     legend(leg, 'Location', 'NorthEastOutside')
94     title_handle = {sprintf('Max velocity for different alpha values and variable
viscosity, '); '0.25 in amplitude and 300Hz in frequency'};
95     title(title_handle, 'FontWeight', 'bold')
96     set(gca, 'ActivePositionProperty', 'OuterPosition');
97     fig_name = sprintf('max_velocity_alpha_visc_summary');
98     print(figure(teller), '-djpeg', fig_name)
99
100    figure(teller+1)
101    plot(alpha_num, max_vel)
102
103    xlabel('Alpha [-]')
104    ylabel('Velocity [m/s]')
105    legend(leg2, 'Location', 'NorthEastOutside')
106    title_handle = {sprintf('Max velocity for different alpha values and variable
viscosity, '); '0.25 in amplitude and 300Hz in frequency'};
107    title(title_handle, 'FontWeight', 'bold')
108    set(gca, 'ActivePositionProperty', 'OuterPosition');
109    fig_name = sprintf('max_velocity_alpha_summary');
110    print(figure(teller+1), '-djpeg', fig_name)
111
112    figure(teller+2)
113    plot(alpha_num, mean_vel_stored)
114
115    xlabel('Alpha [-]')
116    ylabel('Velocity [m/s]')
117    legend(leg2, 'Location', 'NorthEastOutside')
118    title_handle = {sprintf('Mean velocity for different alpha values and variable
viscosity, '); '0.25 in amplitude and 300Hz in frequency'};
119    title(title_handle, 'FontWeight', 'bold')
120    set(gca, 'ActivePositionProperty', 'OuterPosition');
121    fig_name = sprintf('mean_velocity_alpha_summary');
```

```
122         print(figure(teller+2), '-djpeg', fig_name)
123
124
125
```

A.18 Matlab script for autogeneration of simulated force plots.


```
1  clc
2  clear all
3
4  % alpha =
   {sprintf('05'),sprintf('06'),sprintf('07'),sprintf('08'),sprintf('085'),sprintf('09'),spr
   intf('092'),sprintf('094')});
5  alpha = {sprintf('05'),sprintf('06'),sprintf('07'),sprintf('08'),sprintf('085'),sprintf(
   '09'),sprintf('092')});
6  alpha_num = [0.5 0.6 0.7 0.8 0.85 0.9 0.92];
7
8  N_alpha = length(alpha);
9
10 visc = {sprintf('1e-3'),sprintf('9e-4'),sprintf('8e-4'),sprintf('7e-4'),sprintf('6e-4'),
   sprintf('5e-4'),sprintf('4e-4'),sprintf('3e-4'),sprintf('2e-4'),sprintf('1e-4')});
11 visc_num = [1e-3 9e-4 8e-4 7e-4 6e-4 5e-4 4e-4 3e-4 2e-4 1e-4];
12
13 N_visc = length(visc);
14
15 leg = cell(N_alpha,1);
16 leg(:, :) = {' '};
17
18     i_teller = 1;
19     v_teller = 101;
20     teller = 1001;
21
22 for iter=1:N_visc
23
24     for jter=1:N_alpha
25
26         leg(jter, :) = {sprintf('Alpha value of %s ',alpha{jter})};
27
28         temp = sprintf('visc%s_alpha%s/forces/0/forces.dat',visc{iter},alpha{jter});
29
30         fid = fopen(temp,'rt');
31
32         formatSpec = '%f(((%f %f %f) (%f %f %f)) ((%f %f %f) (%f %f %f)))';
33         delimiter_input = ['\t'];
34         C = textscan(fid,formatSpec,'Delimiter',delimiter_input,'CommentStyle',{'#'});
35         fclose(fid);
36
37         %# put columns in separate variables
38         [time,A,B,C,D,E,F,G,H,I,J,K,L] = deal(C{:});
39
40         time = time(350:end);
41         temp_pressure = B(350:end);
42         temp_visc = E(350:end);
43
44         stored_pressure(jter, :) = temp_pressure;
45         stored_visc(jter, :) = temp_visc;
46
47         force_max_pressure(iter,jter) = max(temp_pressure);
48         force_max_visc(iter,jter) = max(temp_visc);
49
50         force_difference = force_max_pressure./force_max_visc;
51
52         hold all
53         figure(teller);
54
55         plot(time,stored_pressure(jter,:),'time,stored_visc(jter,:)')
56
57         xlabel('Time [s]')
58         ylabel('Force per length unit [N/mm]')
59         legend(leg(jter),'Location','NorthEastOutside')
60         title_handle = {sprintf('Pressure and viscous force for alpha value of %s, %s
```

```

    m^2/s in viscosity,' ,alpha{jter},visc{iter});...
61     '0.25 in amplitude and frequency of 300 Hz.' };
62     title(title_handle,'FontWeight','bold')
63     set(gca, 'ActivePositionProperty', 'OuterPosition');
64     fig_name = sprintf('forces_alpha_%s_viscosity_%s',alpha{jter},visc{iter});
65     print(figure(teller),'-djpeg',fig_name)
66
67     hold off
68
69     teller = teller + 1;
70
71     end
72
73     hold all
74     figure(i_teller);
75
76     plot(time,stored_pressure')
77
78     xlabel('Time [s]')
79     ylabel('Force per length unit [N/mm]')
80     legend(leg,'Location','NorthEastOutside')
81     title_handle = {sprintf('Pressure force for different alpha values, %s m^2/s in
viscosity',' ,visc{iter});...
82     '0.25 in amplitude and frequency of 300 Hz.' };
83     title(title_handle,'FontWeight','bold')
84     set(gca, 'ActivePositionProperty', 'OuterPosition');
85     fig_name = sprintf('pressure_force_alpha_viscosity_%s',visc{iter});
86     print(figure(i_teller),'-djpeg',fig_name)
87
88     hold off
89
90     i_teller = i_teller + 1;
91
92     hold all
93
94     figure(v_teller)
95     plot(time,stored_visc')
96     xlabel('Time [s]')
97     ylabel('Force per length unit [N/mm]')
98     legend(leg,'Location','NorthEastOutside')
99     title_handle = {sprintf('Viscous force for different alpha values, %s m^2/s in
viscosity',' ,visc{iter});...
100    '0.25 in amplitude and frequency of 300 Hz.' };
101    title(title_handle,'FontWeight','bold')
102    set(gca, 'ActivePositionProperty', 'OuterPosition');
103    fig_name = sprintf('viscous_force_alpha_viscosity_%s',visc{iter});
104    print(figure(v_teller),'-djpeg',fig_name)
105
106    hold off
107
108    v_teller = v_teller + 1;
109
110    rel_pressure_force_max(iter,:) = force_max_pressure(iter,:)./force_max_pressure(iter
,1);
111    rel_viscous_force_max(iter,:) = force_max_visc(iter,:)./force_max_visc(iter,1);
112    leg2(iter,:) = {sprintf('Viscosity of %s',visc{iter})};
113
114    end
115
116    %% Processing of the force_max variable
117
118    hold all
119    figure(i_teller+1);
120    plot(alpha_num,rel_pressure_force_max')

```

```
121
122 xlabel('Alpha value [-]')
123 ylabel('Relative maximum pressure force [-]')
124 legend(leg2,'Location','NorthEastOutside')
125 title_handle = {'Relative maximum pressure force for different viscosity and alpha
values,';...
'0.25 in amplitude and frequency of 300 Hz.'};
126
127 title(title_handle,'FontWeight','bold')
128 set(gca, 'ActivePositionProperty', 'OuterPosition');
129 fig_name = 'relative_max_pressure_force';
130 print(figure(i_teller+1),'-djpeg',fig_name)
131
132 hold off
133
134 hold all
135 figure(i_teller+2);
136 plot(alpha_num,rel_viscous_force_max')
137
138 xlabel('Alpha value [-]')
139 ylabel('Relative maximum viscous force [-]')
140 legend(leg2,'Location','NorthEastOutside')
141 title_handle = {'Relative maximum viscous force for different viscosity and alpha
values,';...
'0.25 in amplitude and frequency of 300 Hz.'};
142
143 title(title_handle,'FontWeight','bold')
144 set(gca, 'ActivePositionProperty', 'OuterPosition');
145 fig_name = 'relative_max_viscous_force';
146 print(figure(i_teller+2),'-djpeg',fig_name)
147
148 hold off
149
150 hold all
151 figure(i_teller+3);
152 plot(alpha_num,force_difference')
153
154 xlabel('Alpha value [-]')
155 ylabel('Relative force difference [-]')
156 legend(leg2,'Location','NorthEastOutside')
157 title_handle = {'Relative force difference between pressure and viscous force,';...
'viscosity and alpha values are variable,';...
'0.25 in amplitude and frequency of 300 Hz.'};
158
159 title(title_handle,'FontWeight','bold')
160 set(gca, 'ActivePositionProperty', 'OuterPosition');
161 fig_name = 'relative_force_difference';
162 print(figure(i_teller+3),'-djpeg',fig_name)
163
164 hold off
165
166
167 %%
```


Appendix B

Intermediate velocity plots from case with different viscosity and alpha value.

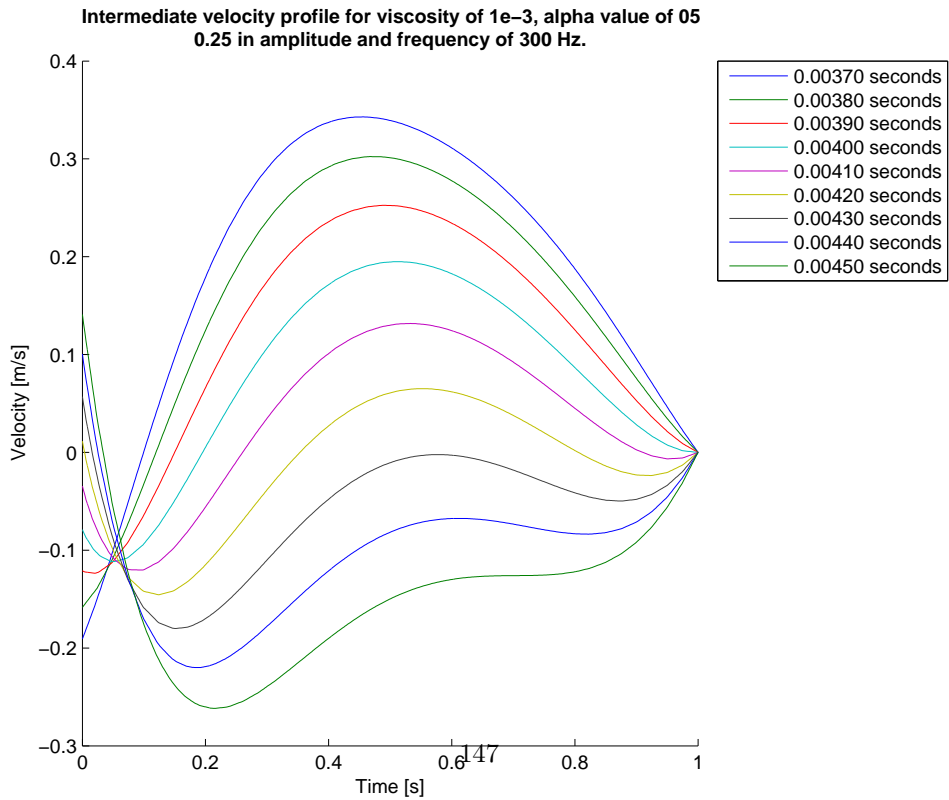


Figure B.1: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.5.

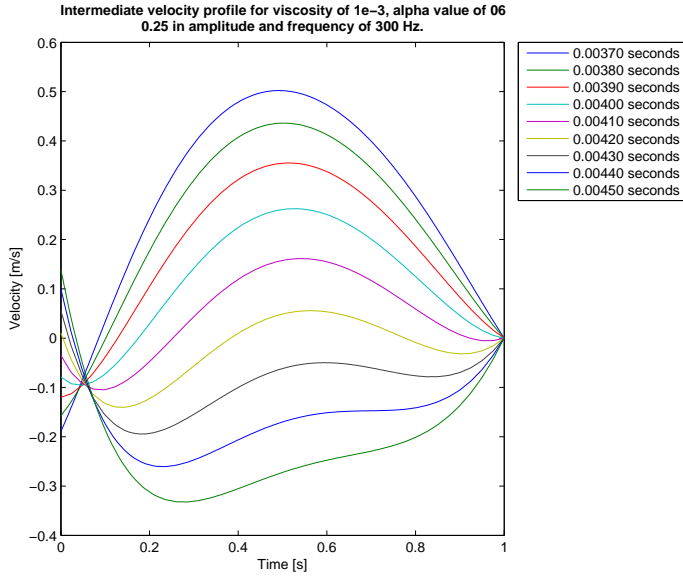


Figure B.2: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.6.

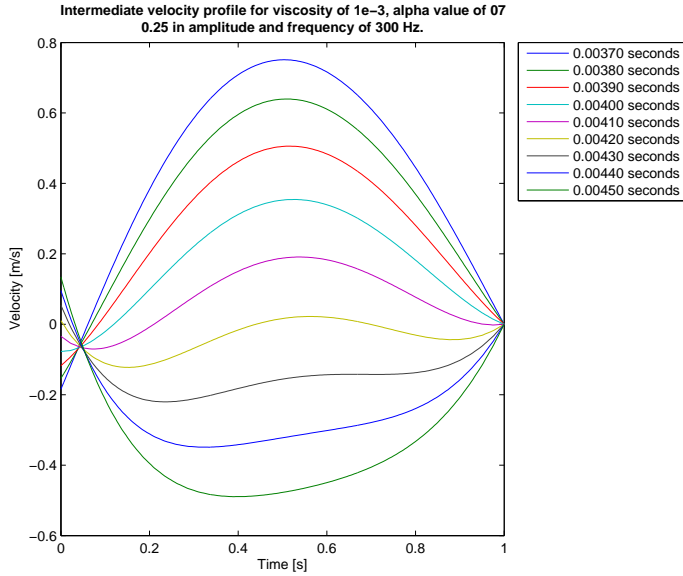


Figure B.3: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.7.

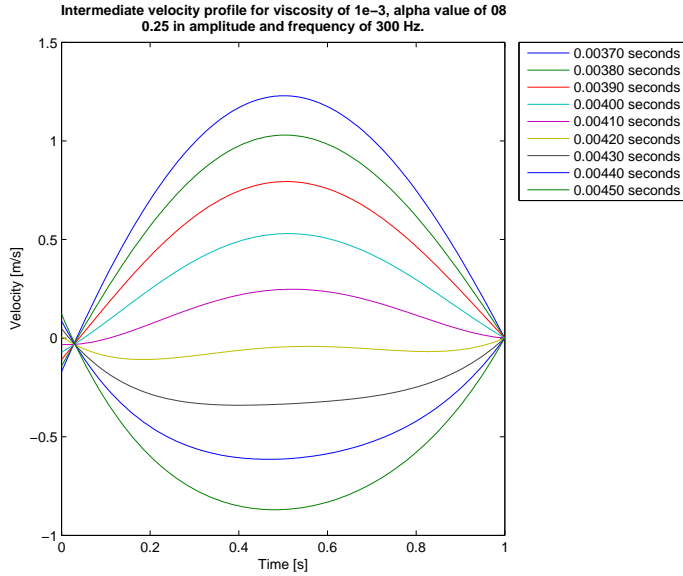


Figure B.4: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.8.

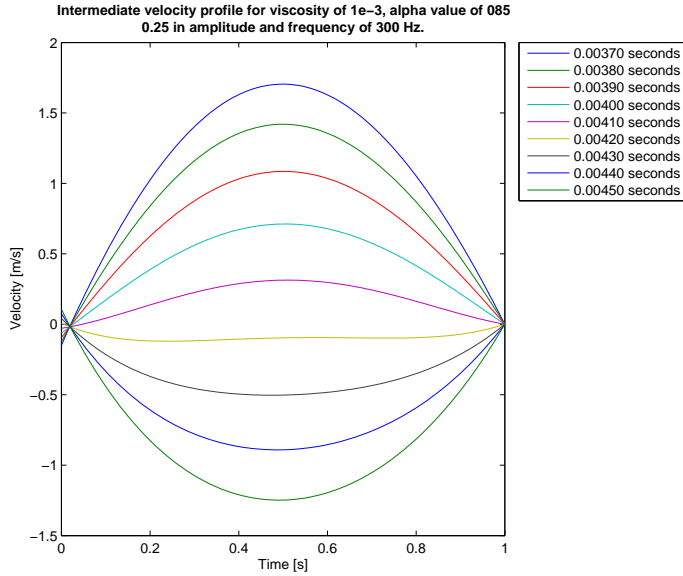


Figure B.5: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.85.

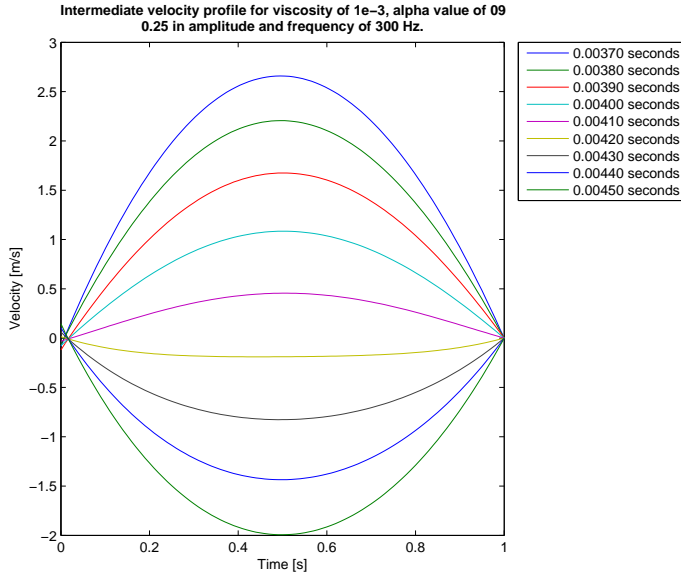


Figure B.6: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.9.

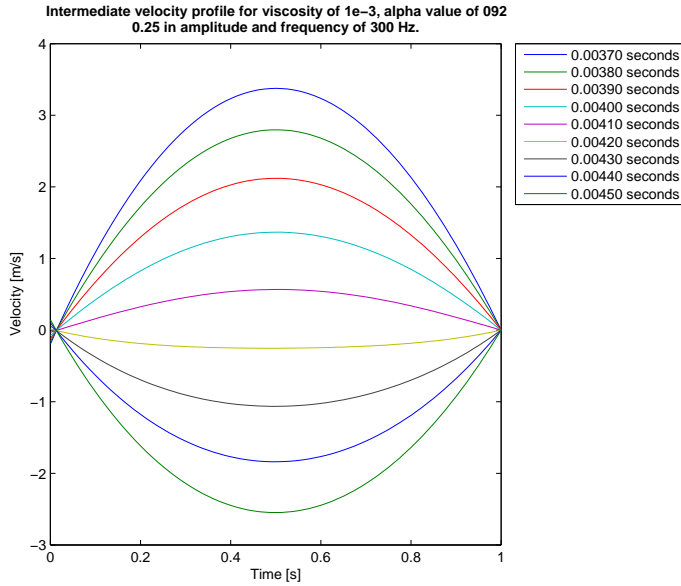


Figure B.7: Case with viscosity of $1e-3 \frac{m^2}{s}$ and alpha value of 0.92.

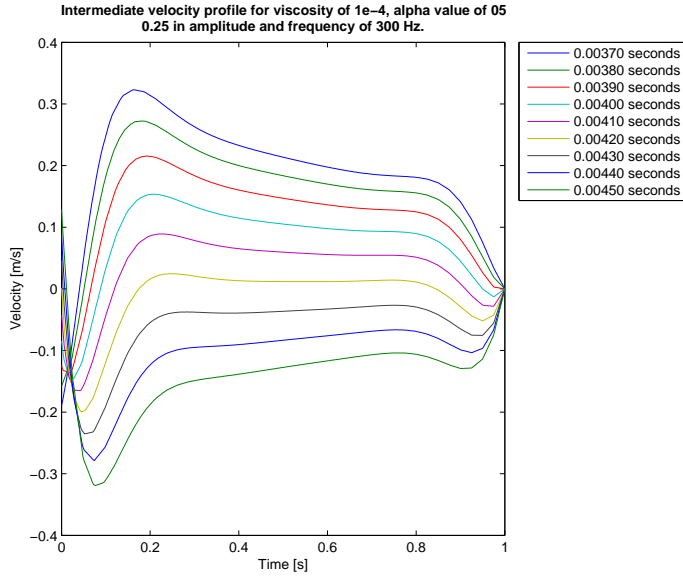


Figure B.8: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.5.

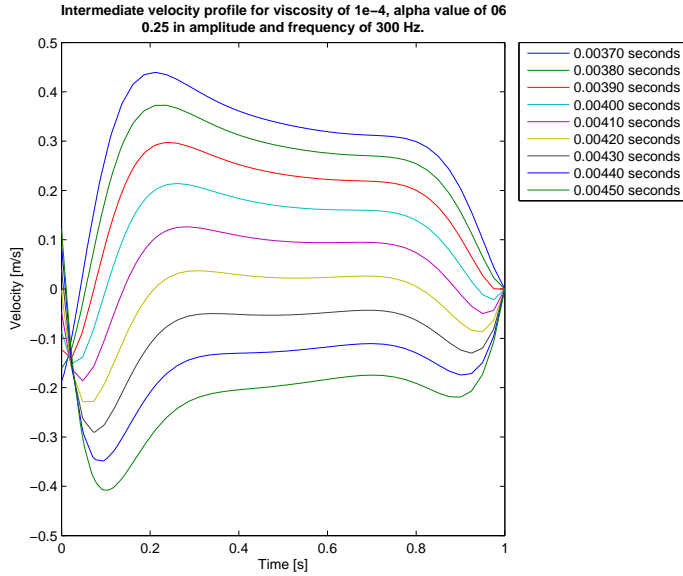


Figure B.9: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.6.

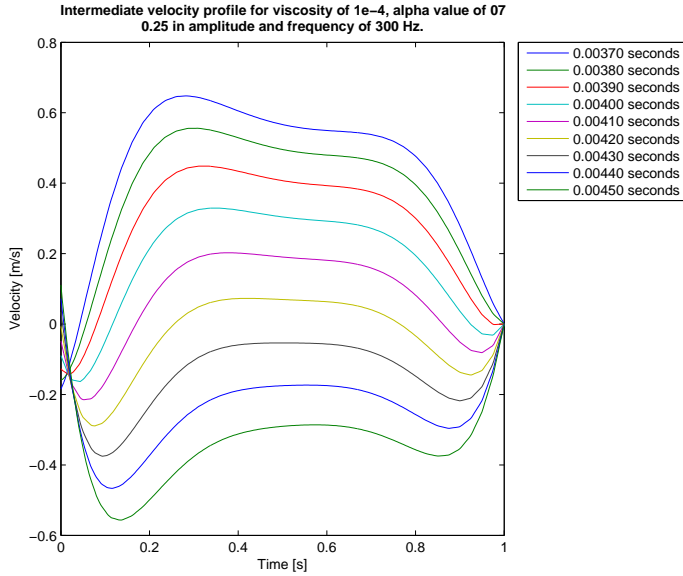


Figure B.10: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.7.

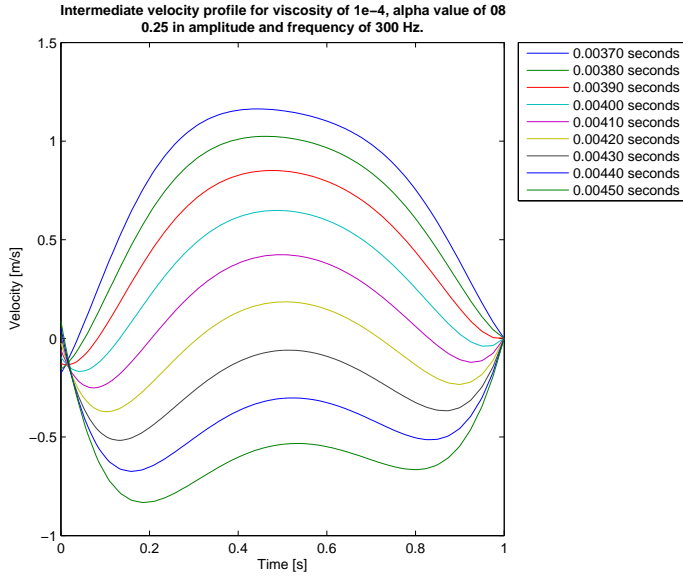


Figure B.11: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.8.

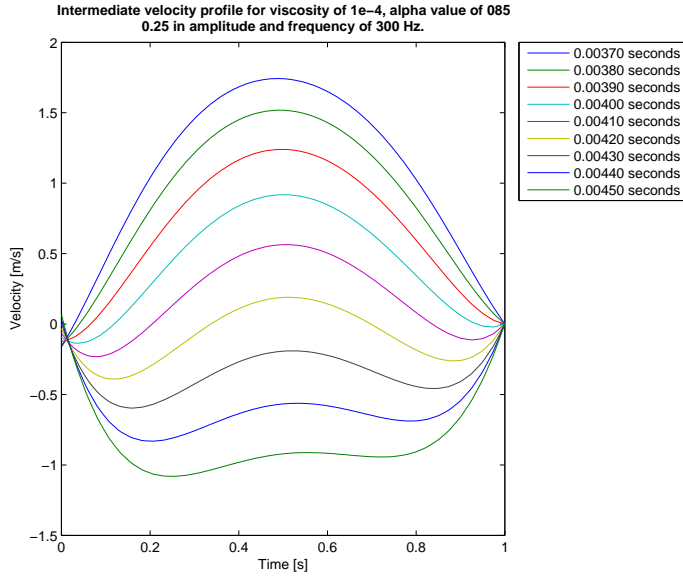


Figure B.12: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.85.

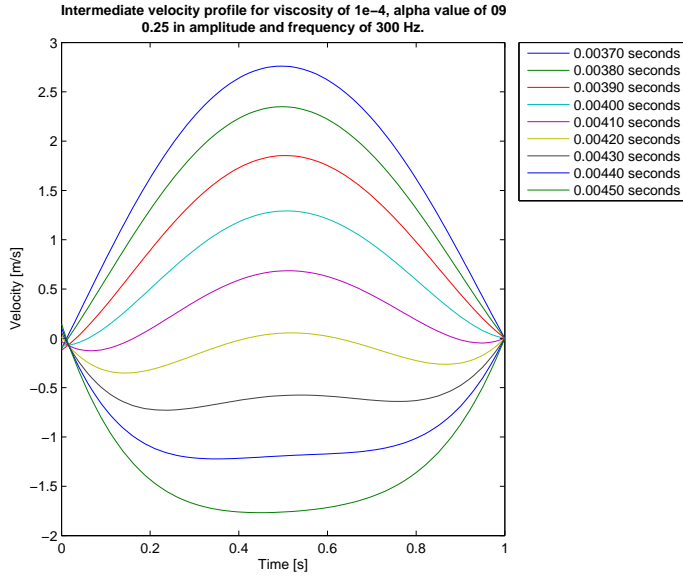


Figure B.13: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.9.

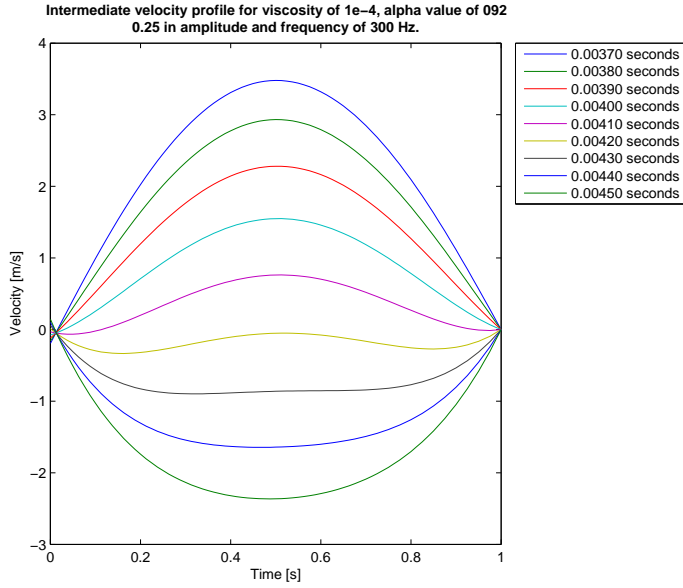


Figure B.14: Case with viscosity of $1e-4 \frac{m^2}{s}$ and alpha value of 0.92.

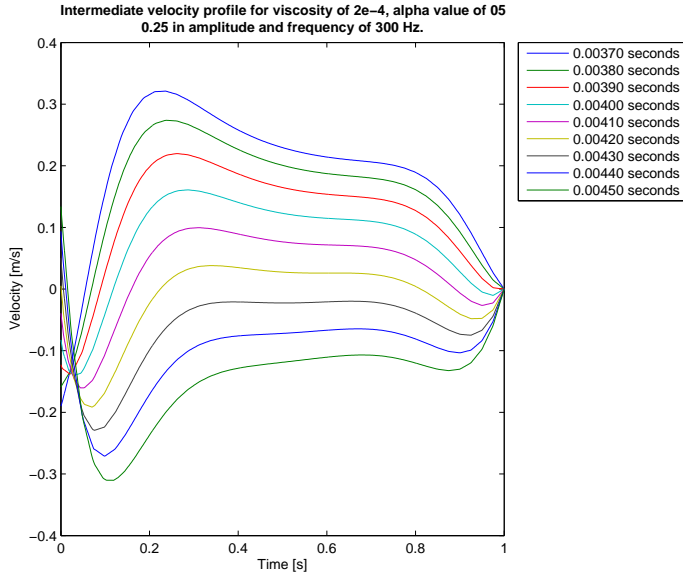


Figure B.15: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.5.

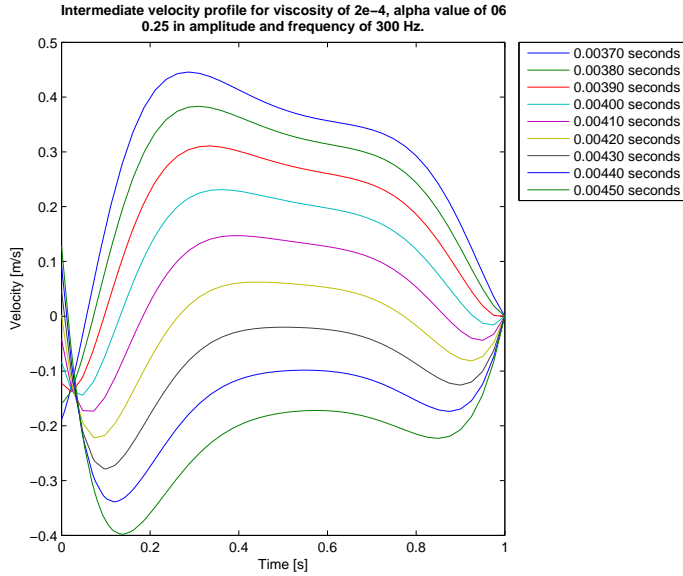


Figure B.16: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.6.

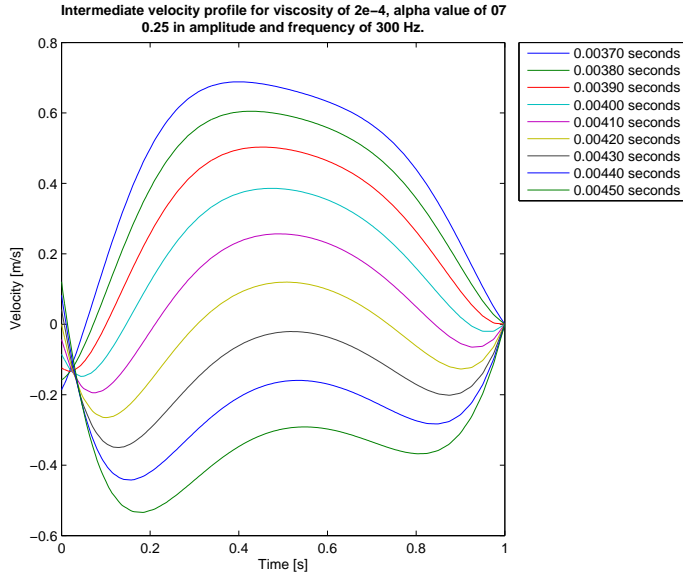


Figure B.17: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.7.

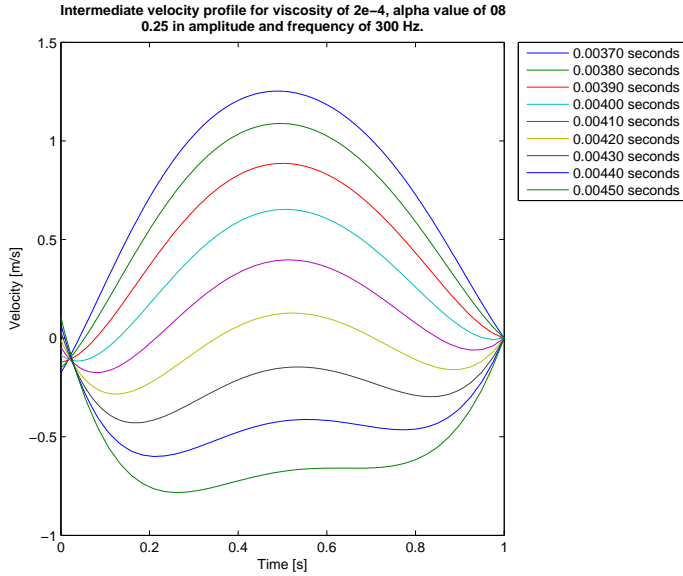


Figure B.18: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.8.

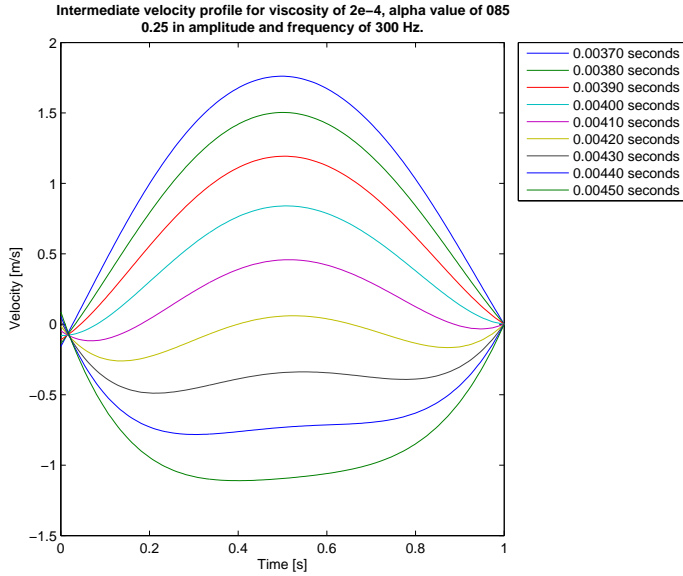


Figure B.19: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.85.

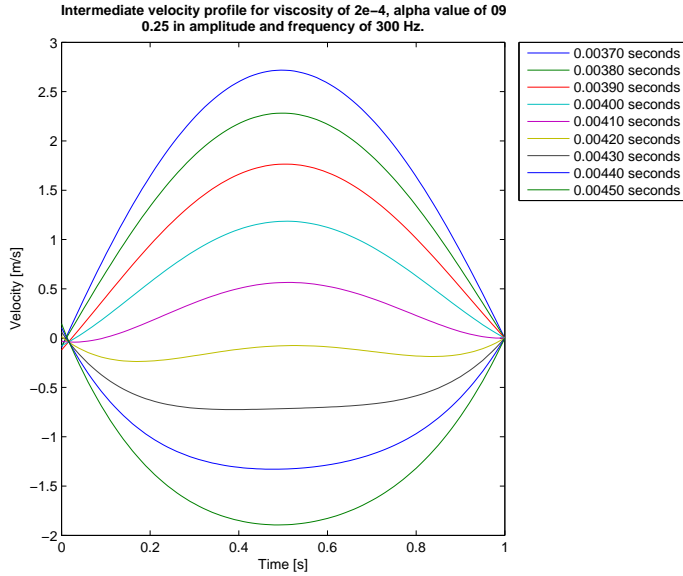


Figure B.20: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.9.

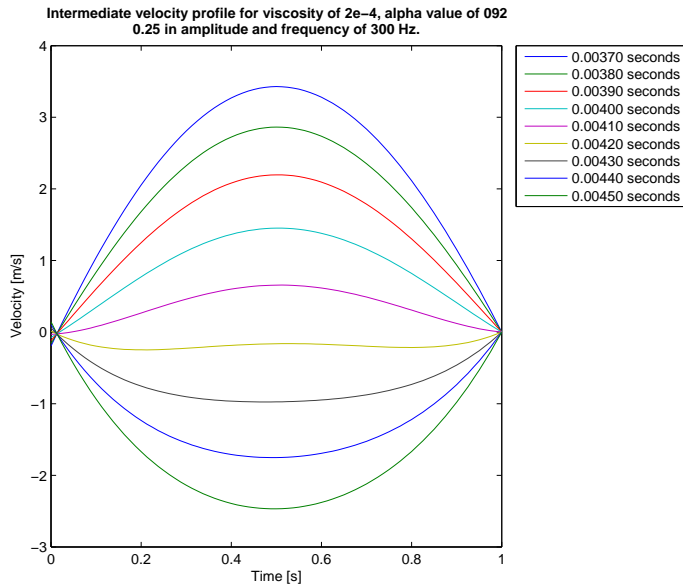


Figure B.21: Case with viscosity of $2e-4 \frac{m^2}{s}$ and alpha value of 0.92.

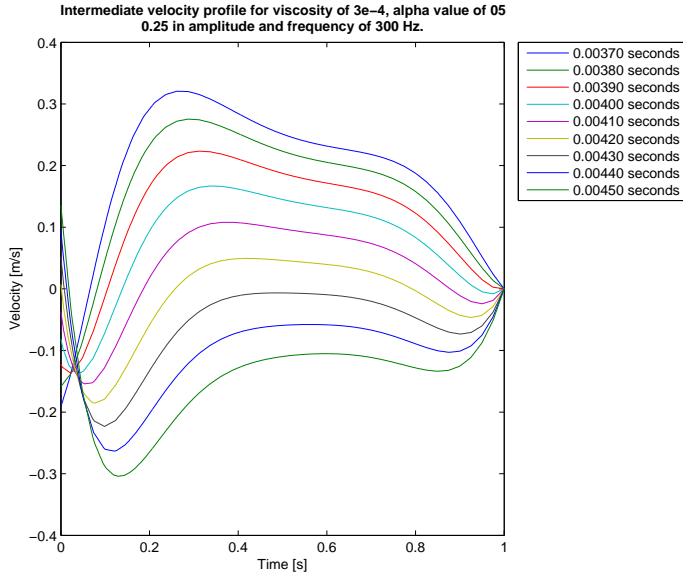


Figure B.22: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.5.

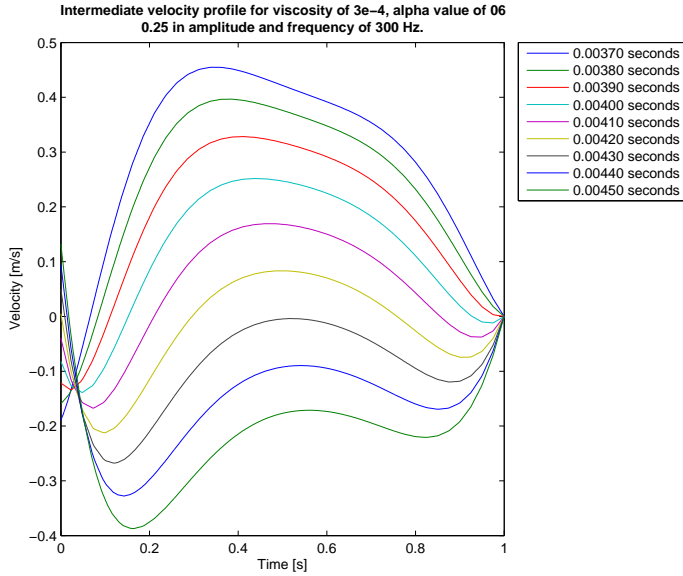


Figure B.23: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.6.

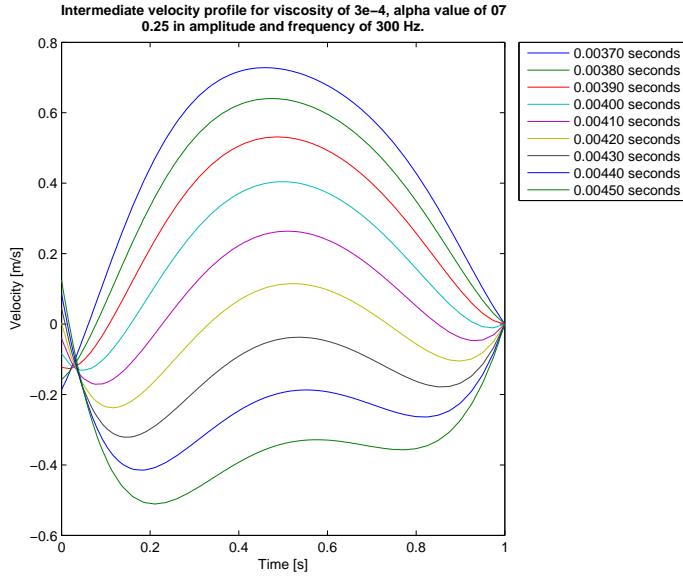


Figure B.24: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.7.

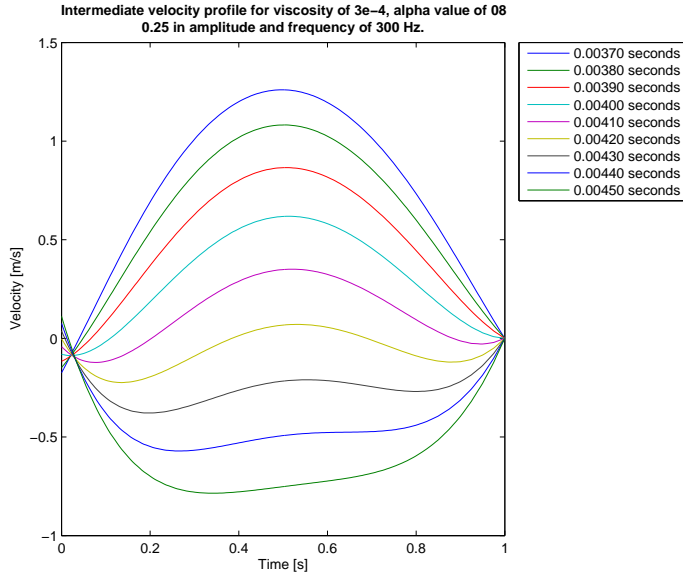


Figure B.25: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.8.

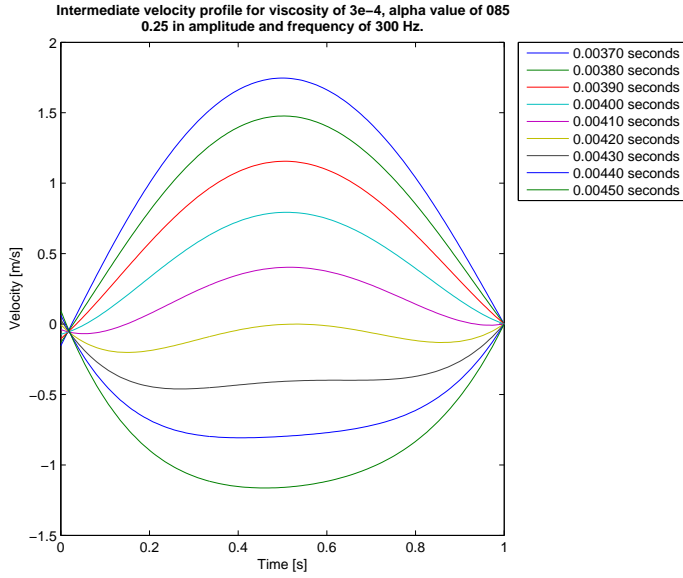


Figure B.26: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.85.

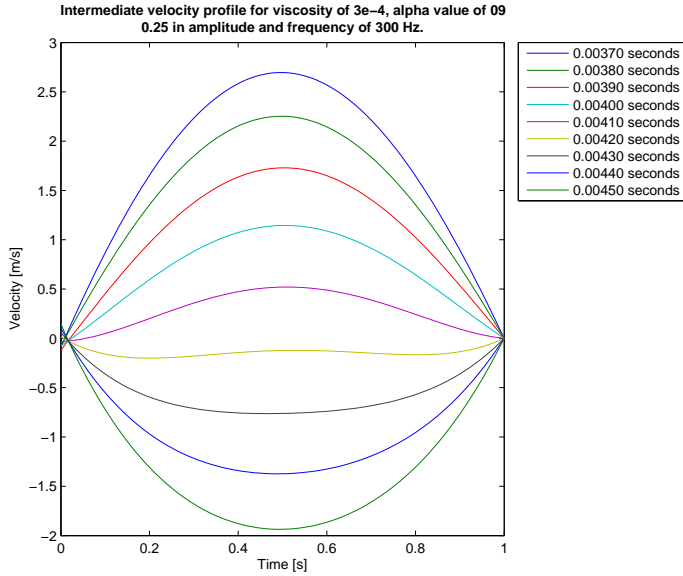


Figure B.27: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.9.

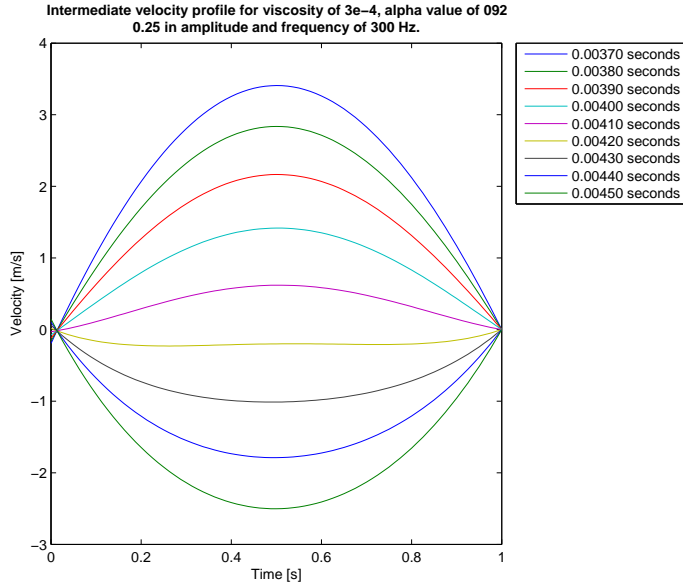


Figure B.28: Case with viscosity of $3e-4 \frac{m^2}{s}$ and alpha value of 0.92.

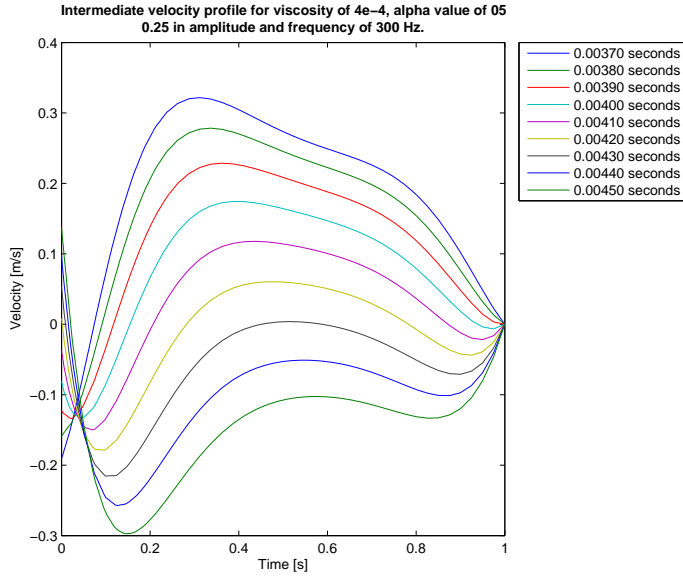


Figure B.29: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.5.

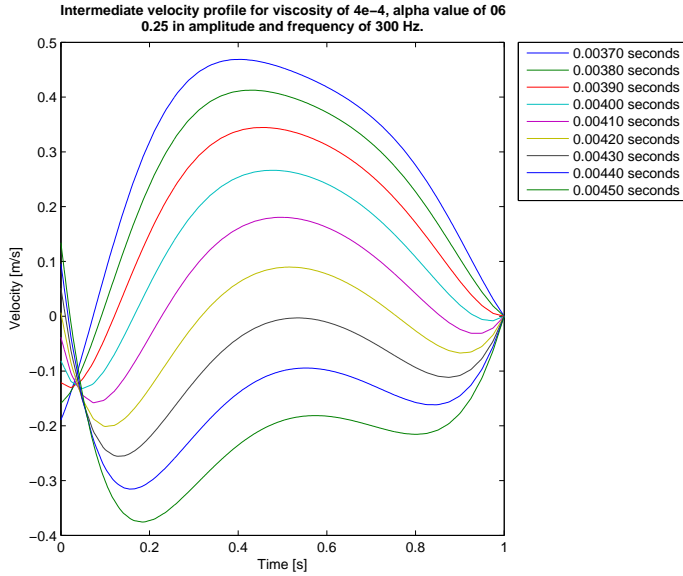


Figure B.30: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.6.

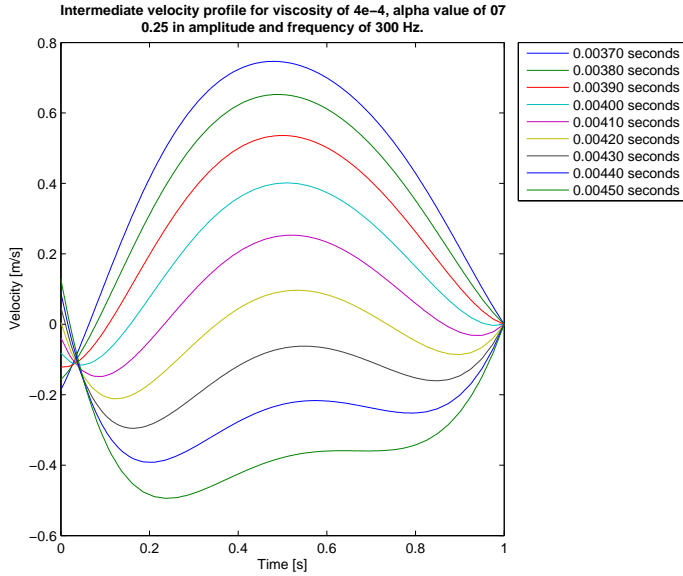


Figure B.31: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.7.

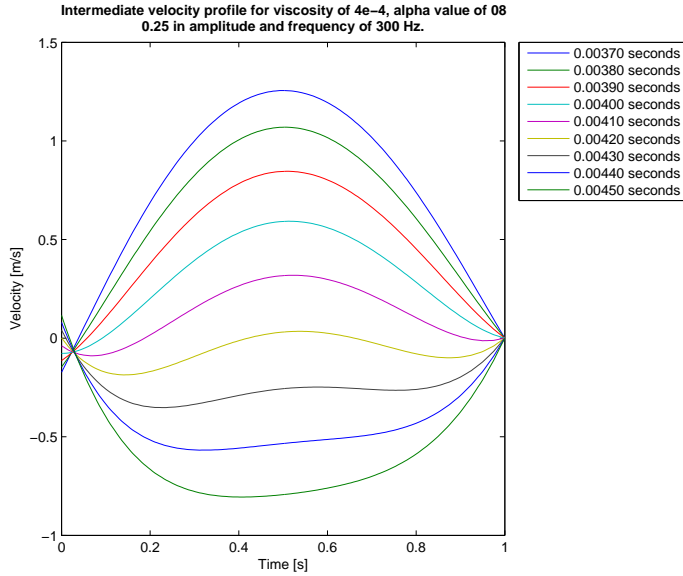


Figure B.32: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.8.

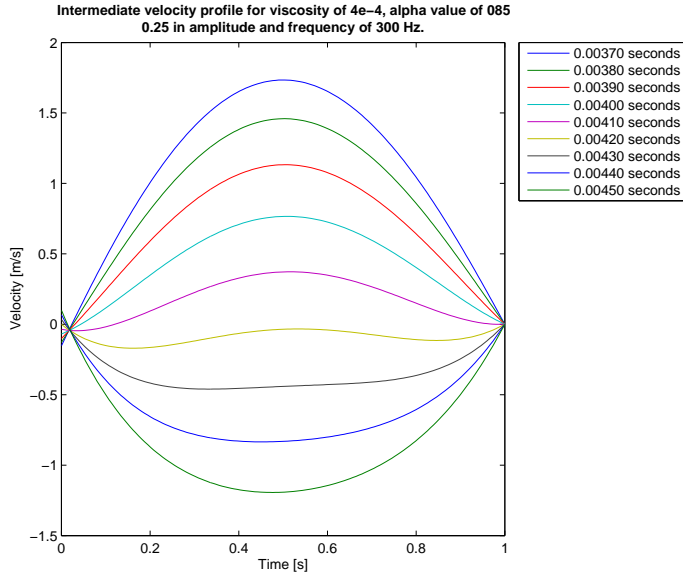


Figure B.33: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.85.

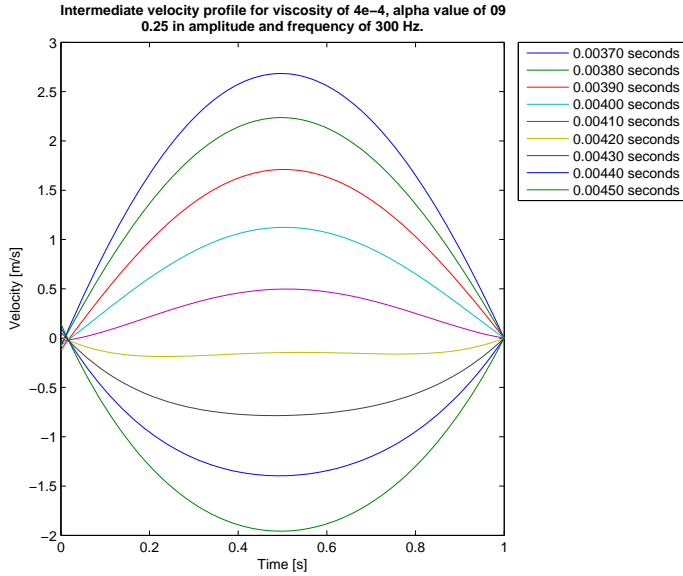


Figure B.34: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.9.

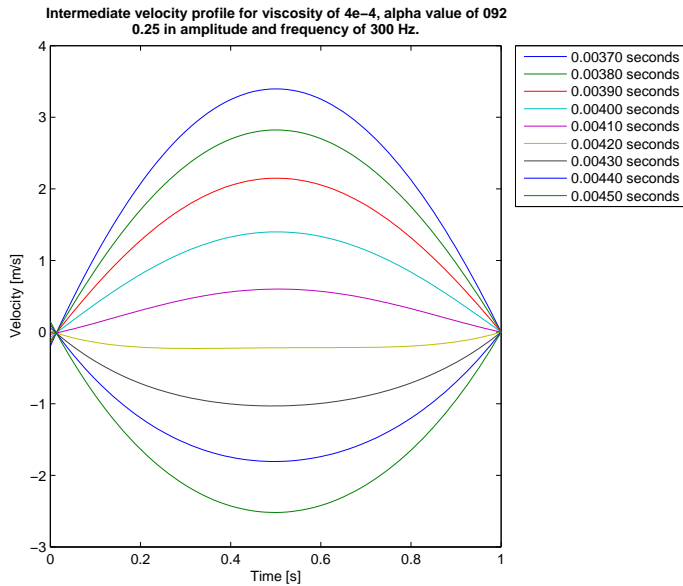


Figure B.35: Case with viscosity of $4e-4 \frac{m^2}{s}$ and alpha value of 0.92.

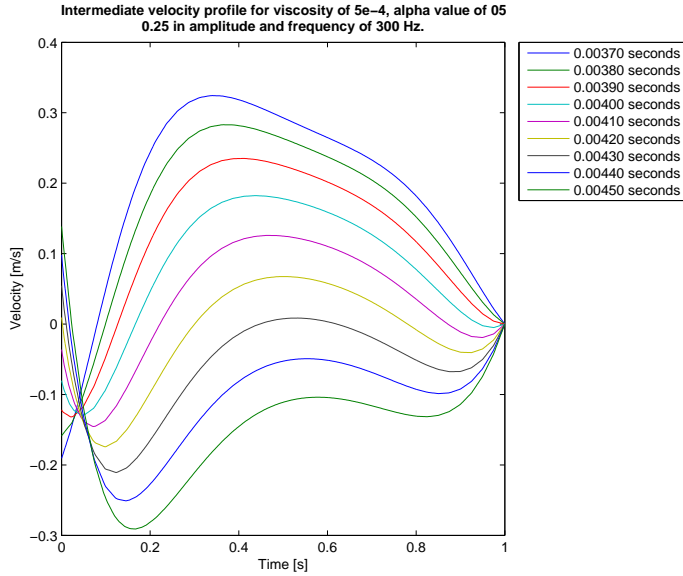


Figure B.36: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.5.

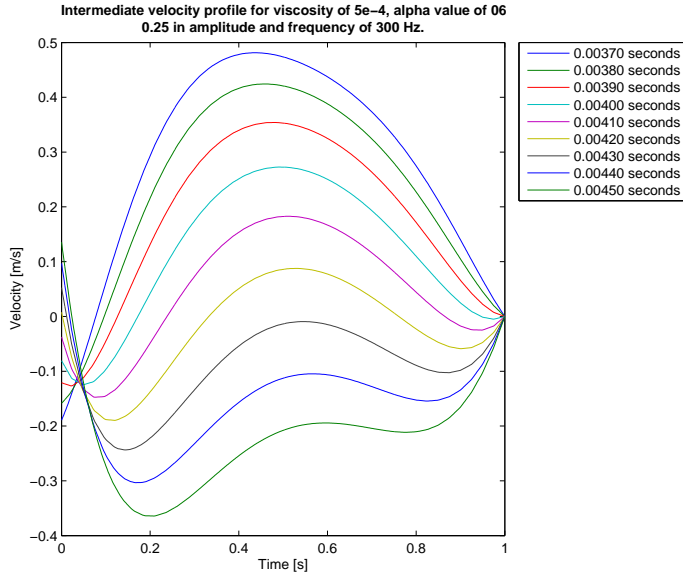


Figure B.37: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.6.

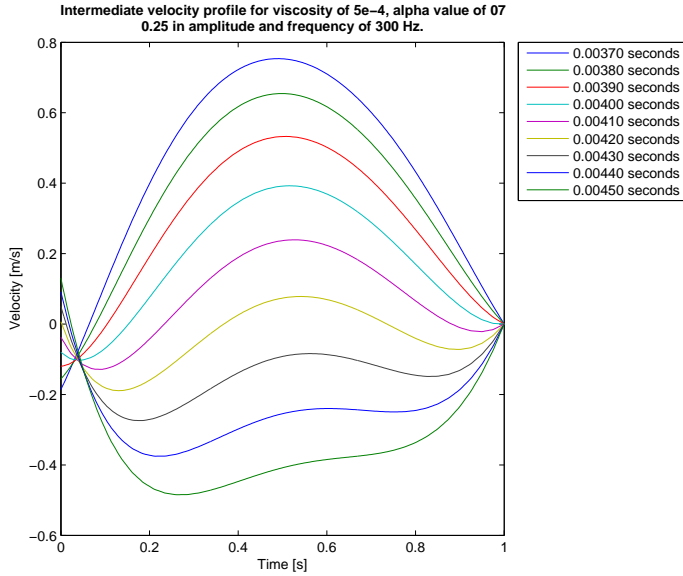


Figure B.38: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.7.

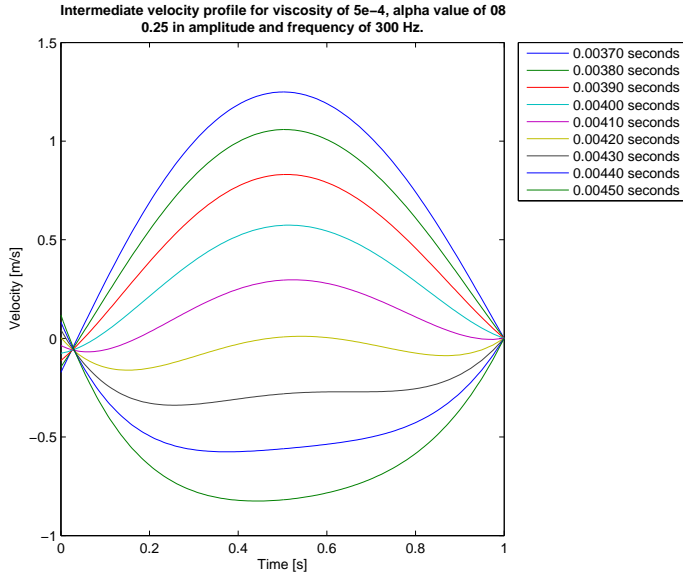


Figure B.39: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.8.

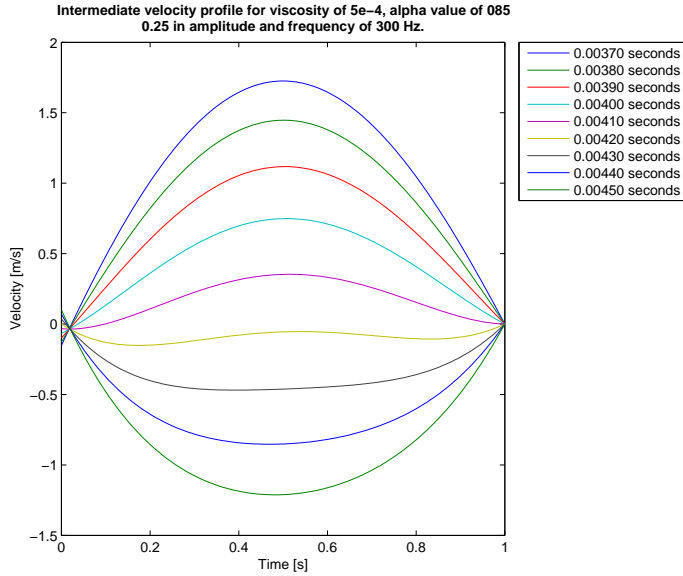


Figure B.40: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.85.

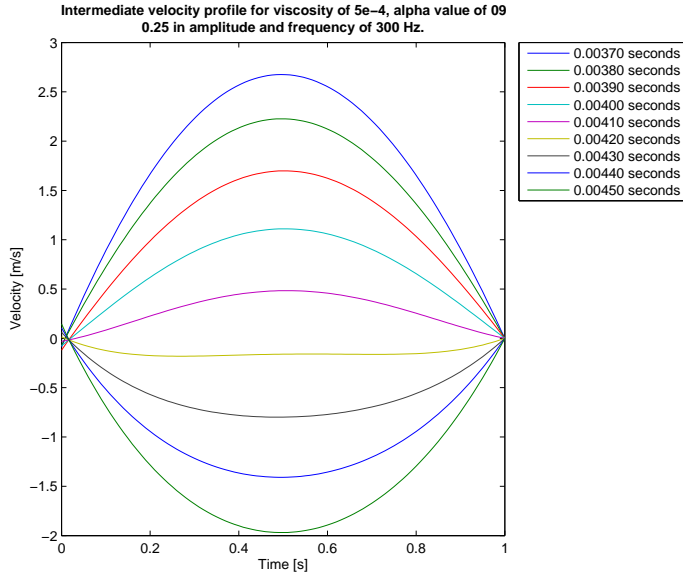


Figure B.41: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.9.

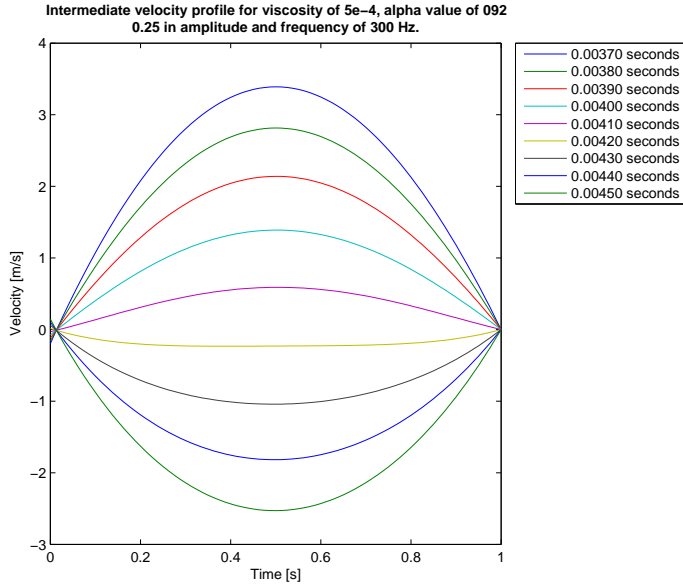


Figure B.42: Case with viscosity of $5e-4 \frac{m^2}{s}$ and alpha value of 0.92.

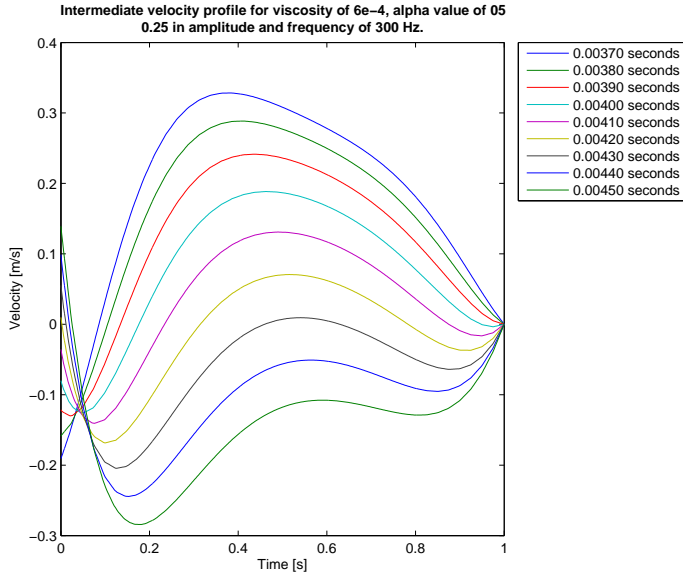


Figure B.43: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.5.

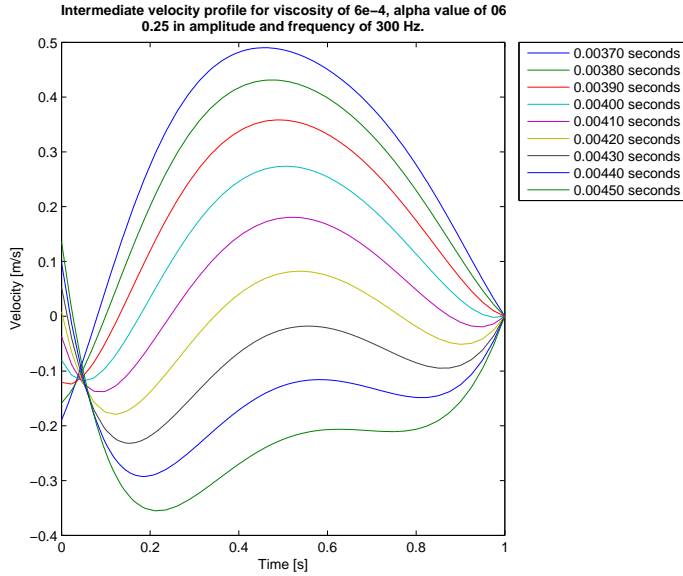


Figure B.44: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.6.

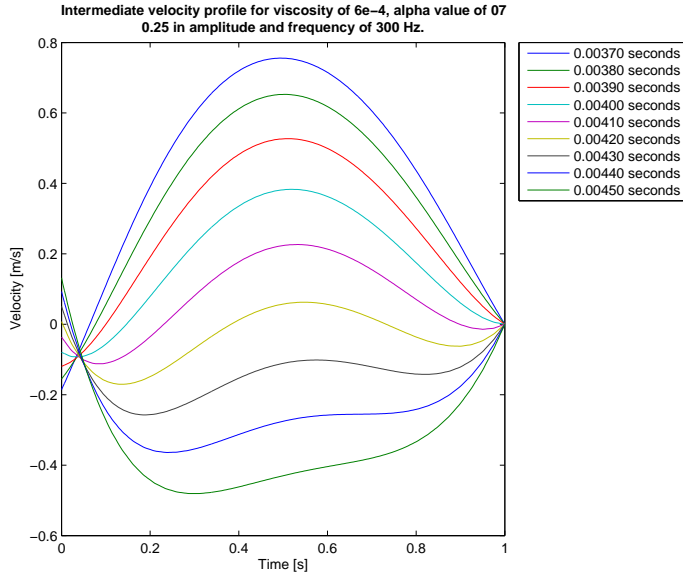


Figure B.45: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.7.

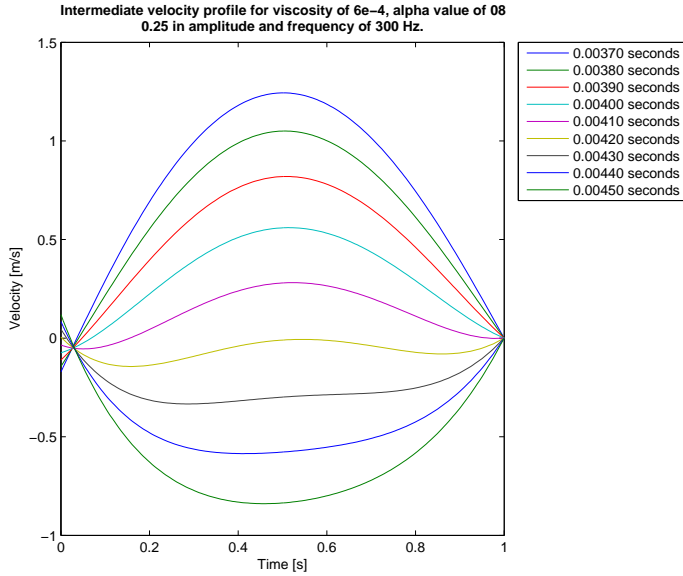


Figure B.46: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.8.

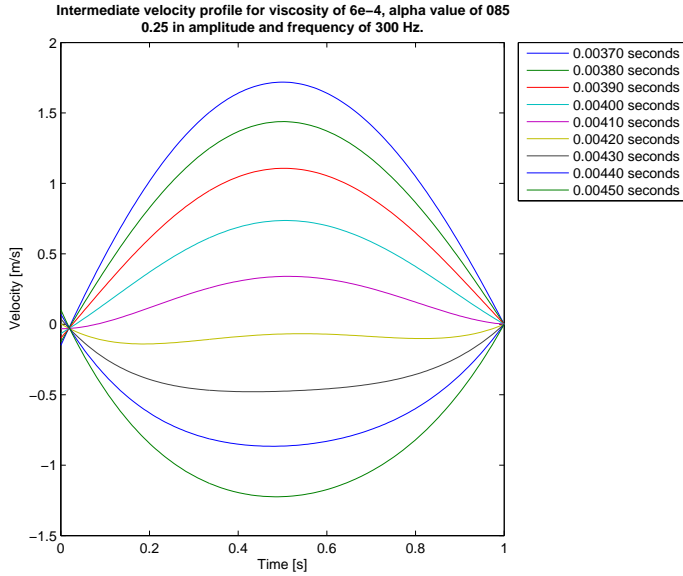


Figure B.47: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.85.

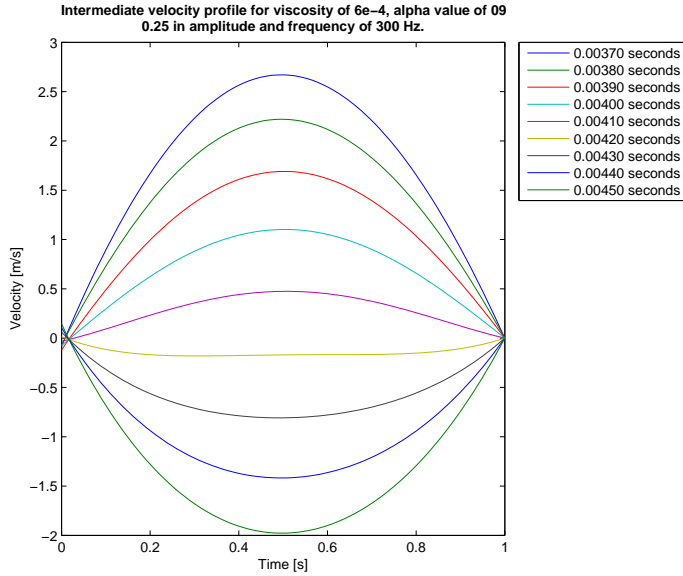


Figure B.48: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.9.

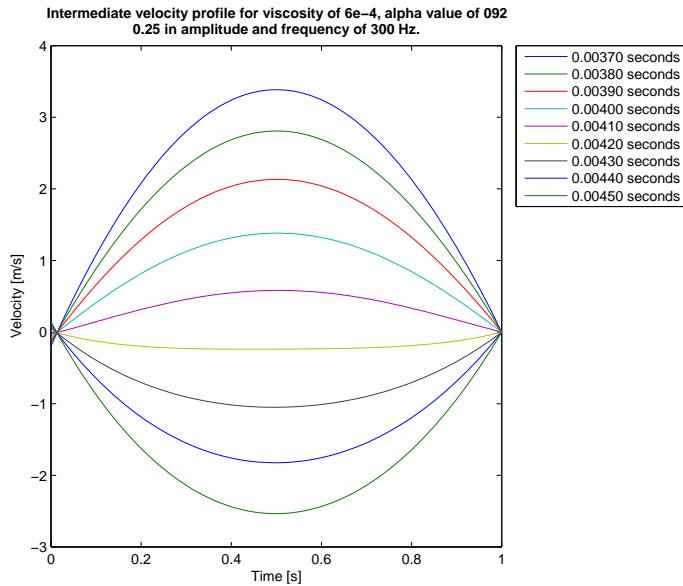


Figure B.49: Case with viscosity of $6e-4 \frac{m^2}{s}$ and alpha value of 0.92.

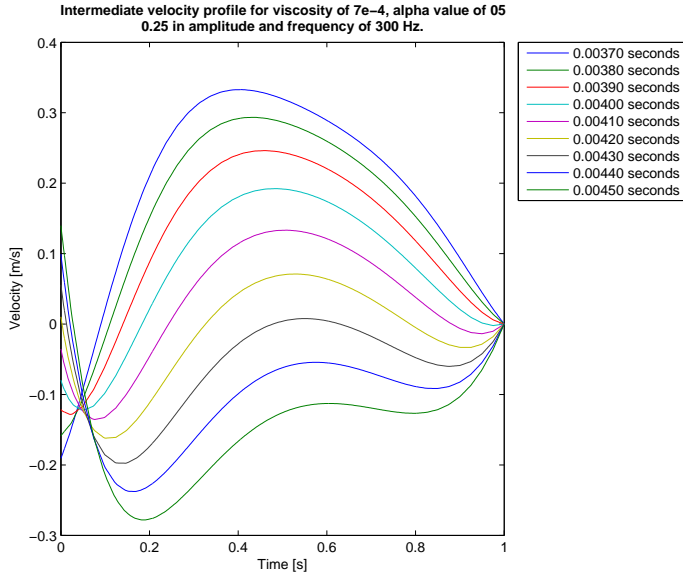


Figure B.50: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.5.

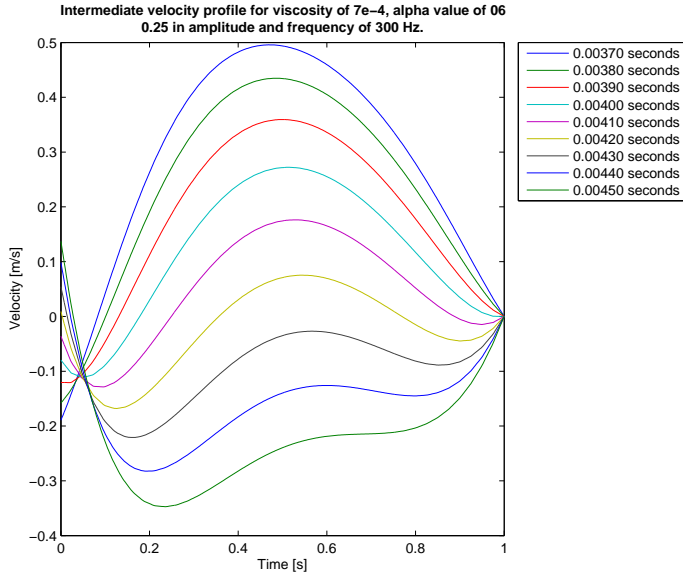


Figure B.51: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.6.

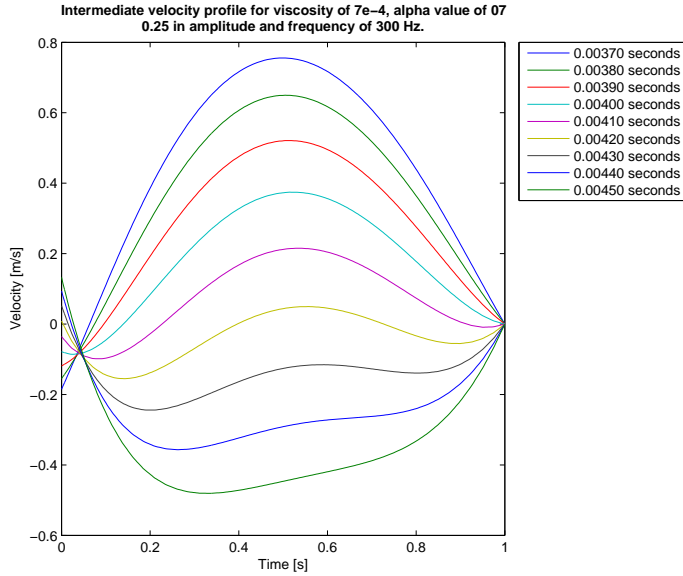


Figure B.52: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.7.

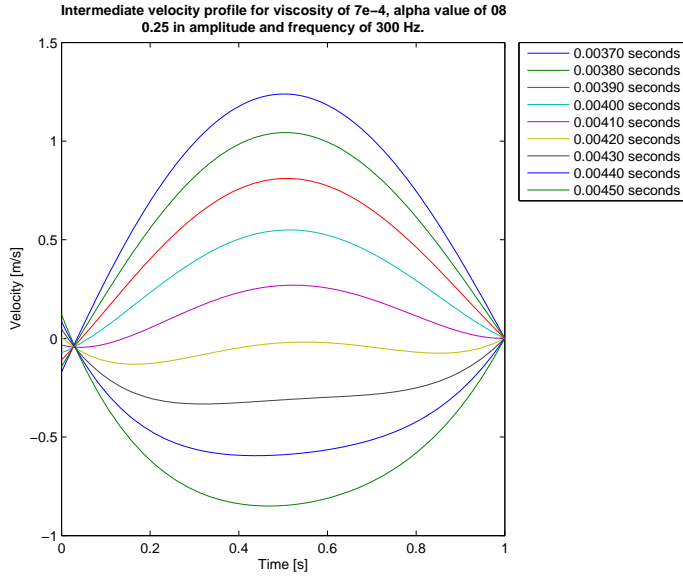


Figure B.53: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.8.

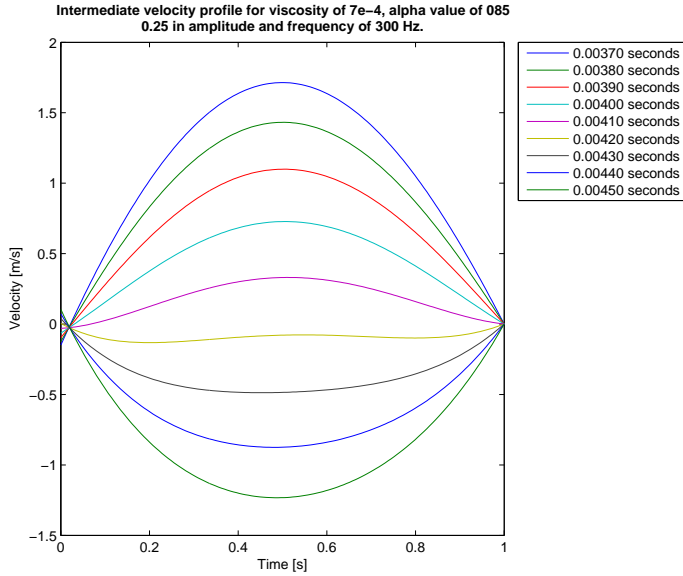


Figure B.54: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.85.

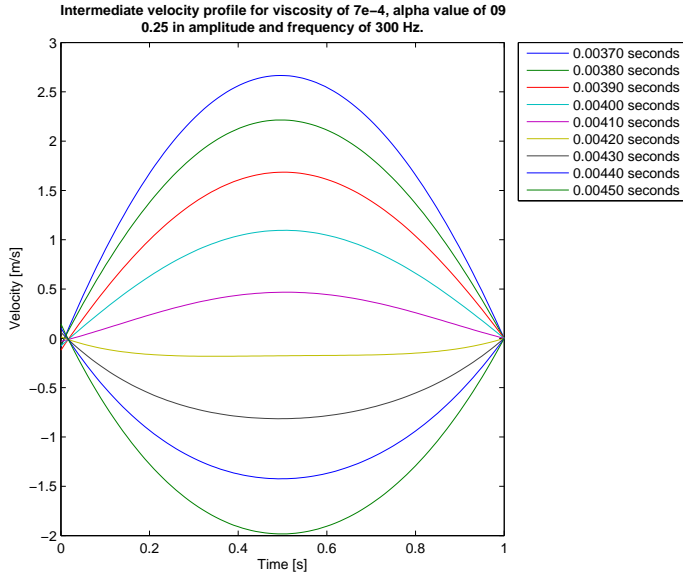


Figure B.55: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.9.

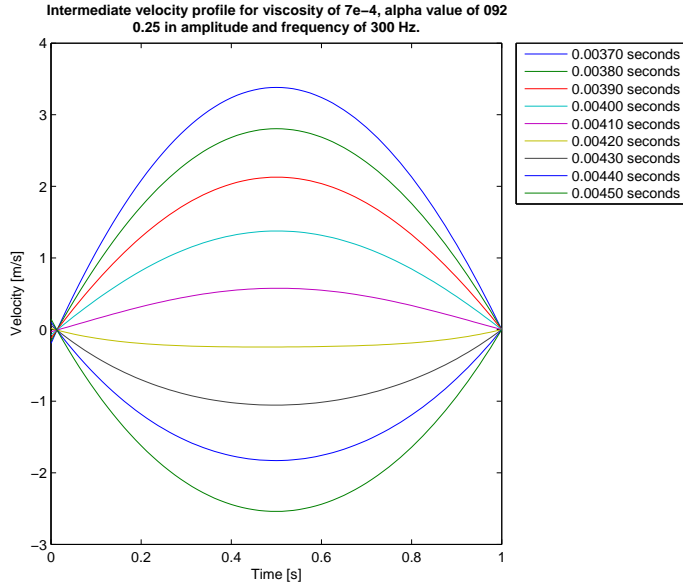


Figure B.56: Case with viscosity of $7e-4 \frac{m^2}{s}$ and alpha value of 0.92.

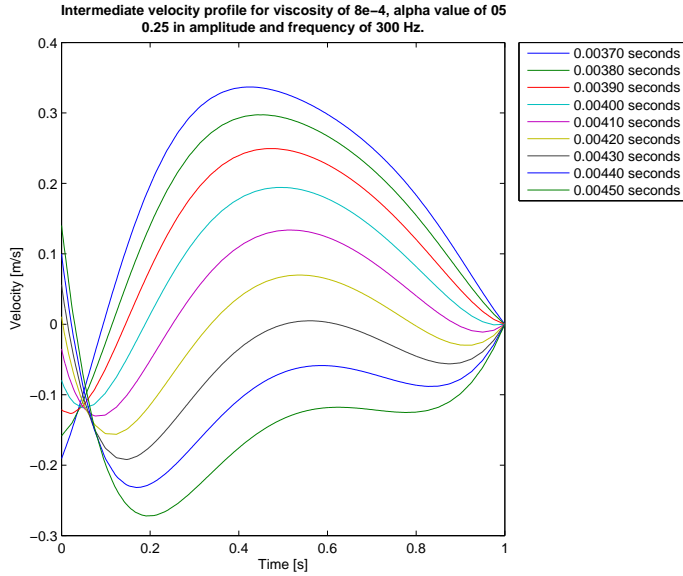


Figure B.57: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.5.

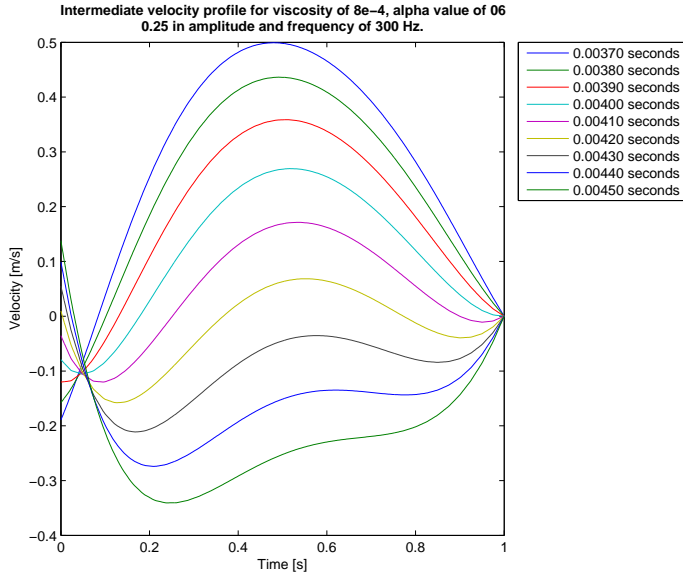


Figure B.58: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.6.

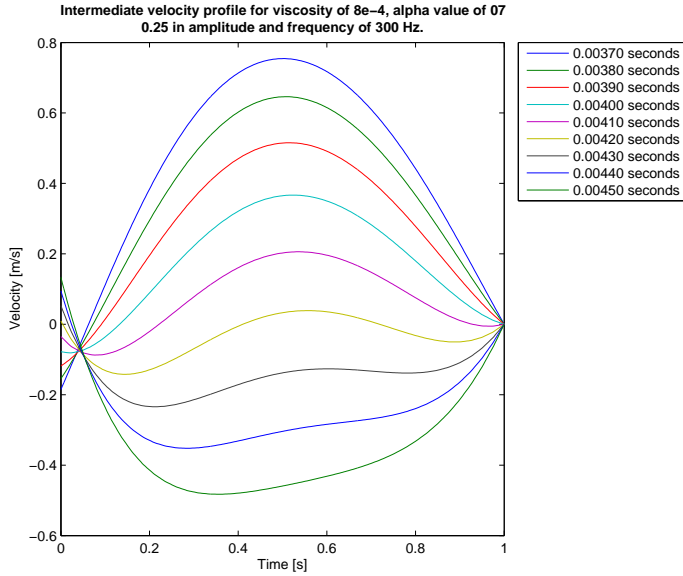


Figure B.59: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.7.

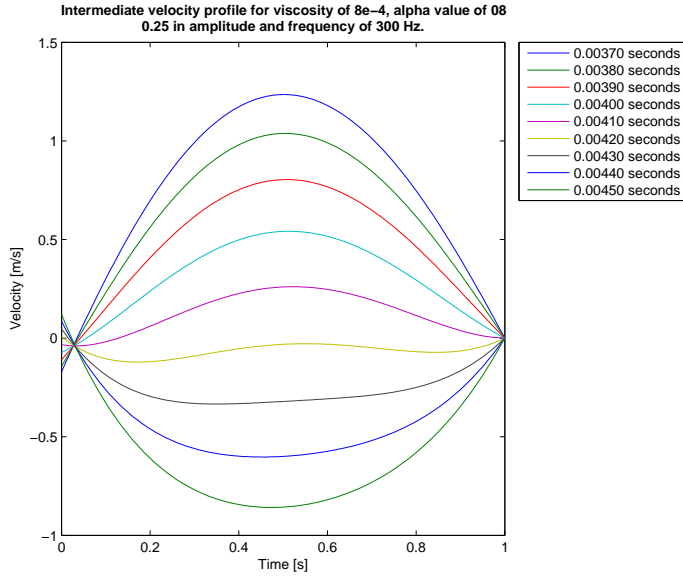


Figure B.60: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.8.

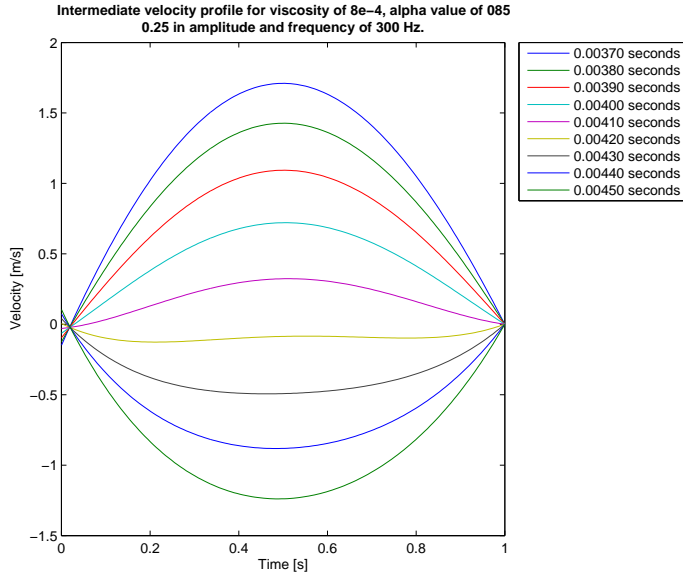


Figure B.61: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.85.

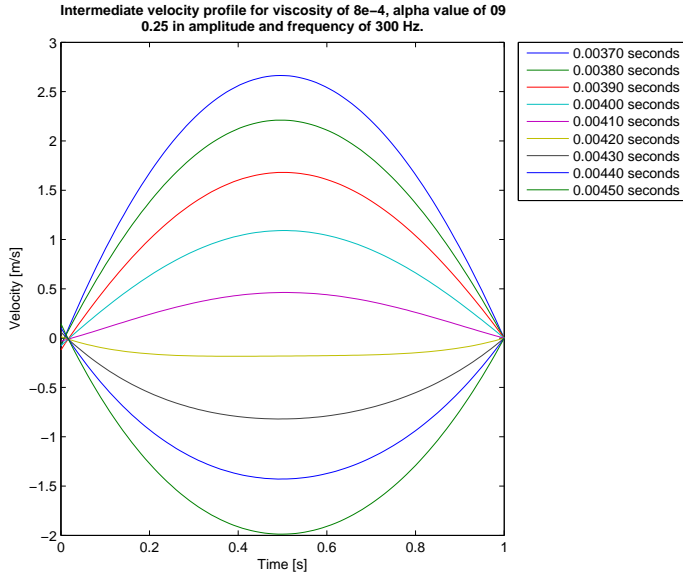


Figure B.62: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.9.

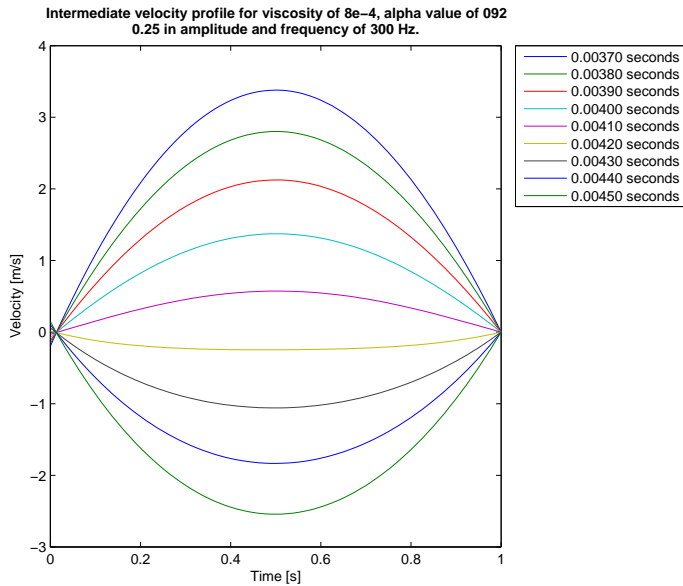


Figure B.63: Case with viscosity of $8e-4 \frac{m^2}{s}$ and alpha value of 0.92.

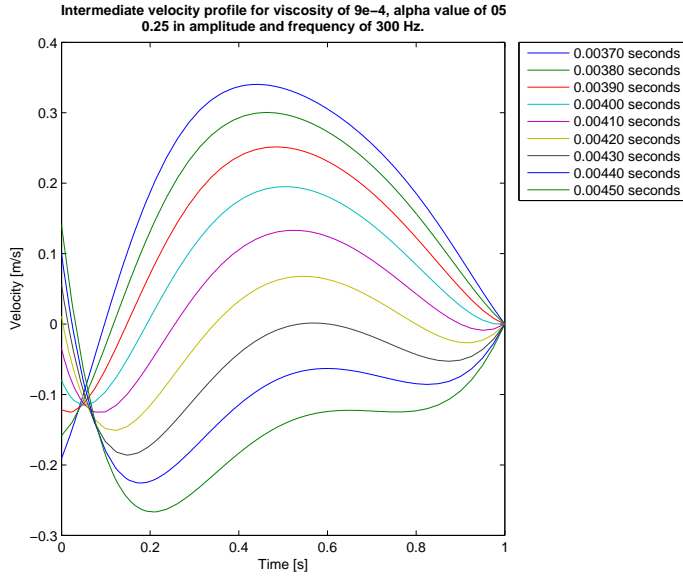


Figure B.64: Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.5.

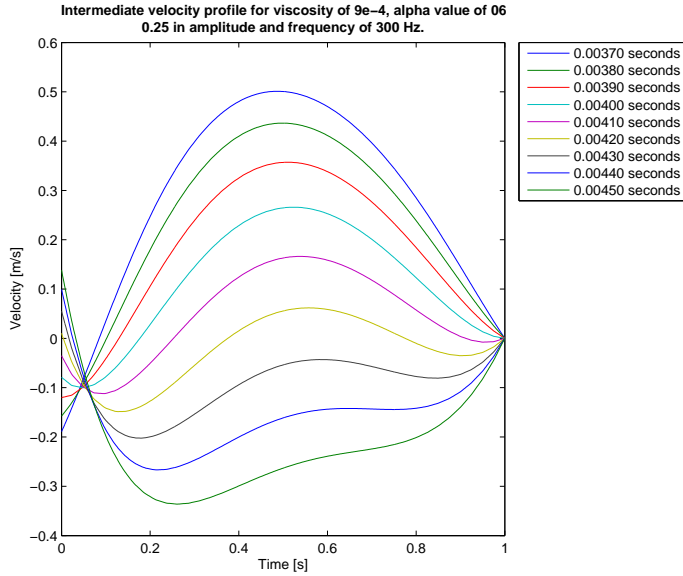


Figure B.65: Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.6.

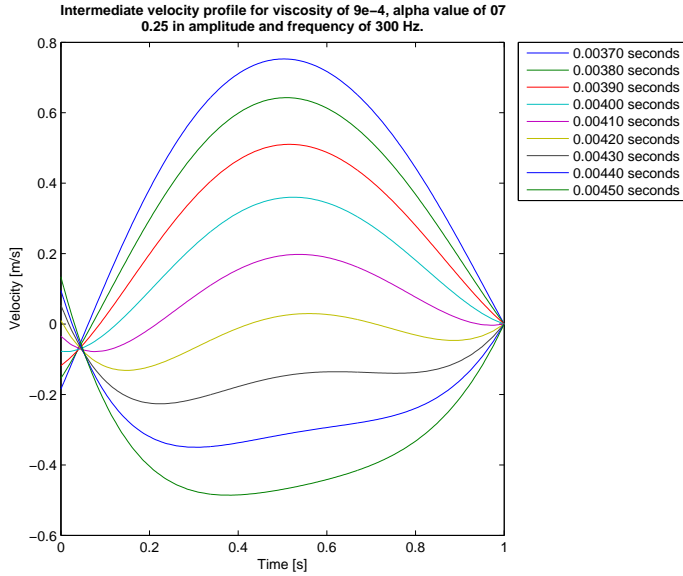


Figure B.66: Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.7.

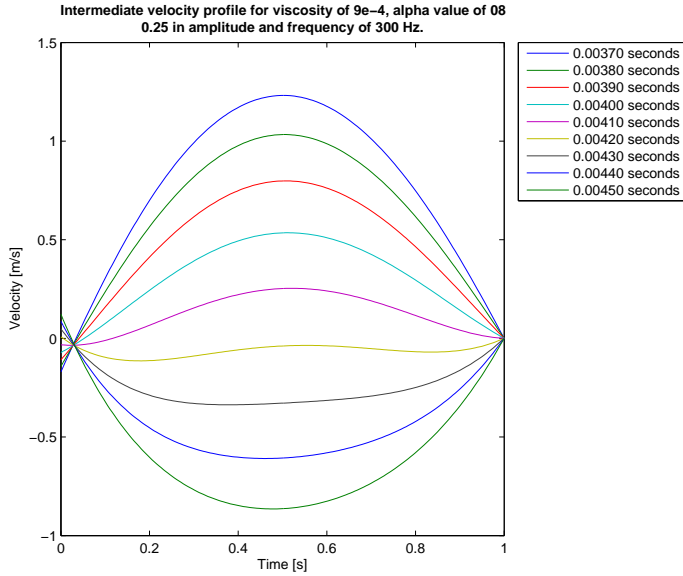


Figure B.67: Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.8.

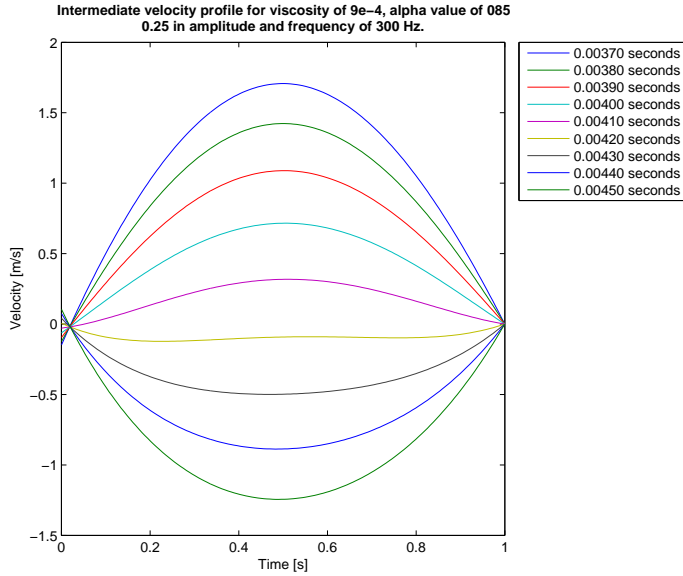


Figure B.68: Case with viscosity of $9\text{e-}4 \frac{m^2}{s}$ and alpha value of 0.85.

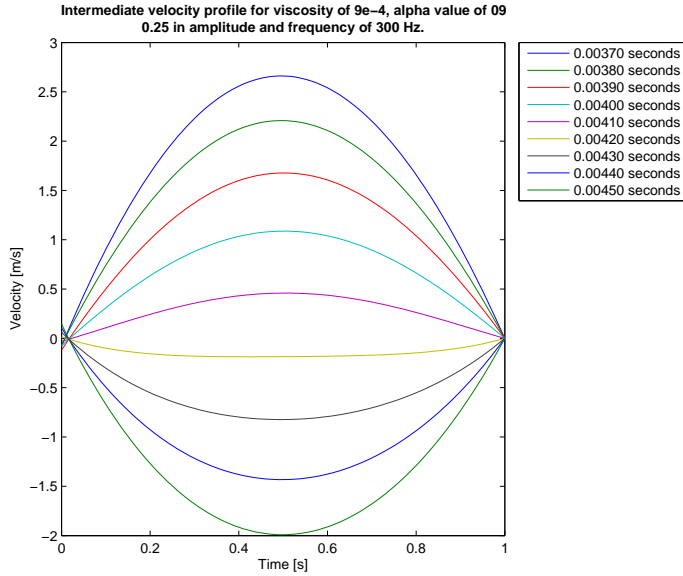


Figure B.69: Case with viscosity of $9\text{e-}4 \frac{m^2}{s}$ and alpha value of 0.9.

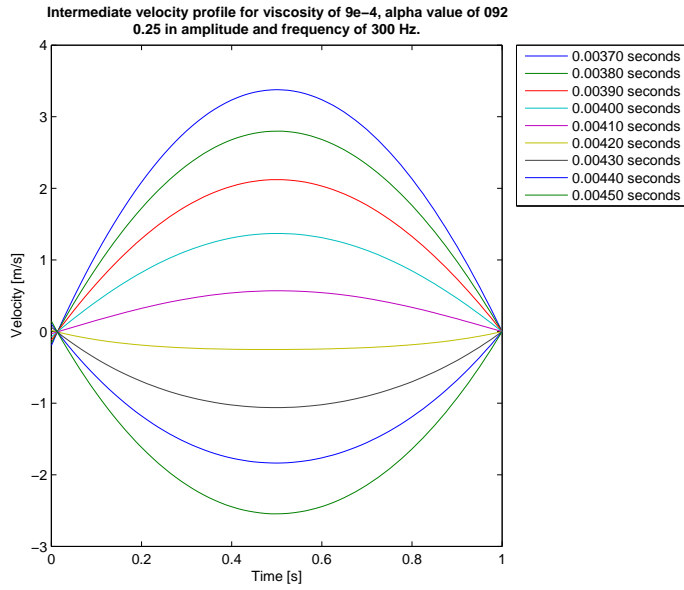


Figure B.70: Case with viscosity of $9e-4 \frac{m^2}{s}$ and alpha value of 0.92.