Drift-flux model usage for steady influx simulation

Ivan Nepomnyashchikh*

Experimental Fluid Mechanics Research Lab, School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, OR, United States

Abstract

This report aims to develop an approach to steady influx modeling using drift-flux model (DFM). First, fundamental mathematical description of phase occurrence/vanishing is presented. Then it is used to model a simplified case of steady air influx into a vertical steady flow of water. Along with that, mathematical analysis of drift-flux model (DFM) is done and a conclusion is made that convergence of the numerical scheme is dependent upon the slip relation parameters. Predicted point of loss of convergence is verified against computer simulation. The resultant software package is created with accordance to [1]

Keywords: DFM, Drif-flux model, steady influx

1. Introduction

Motivation: global goal. Subsurface reservoirs with highly pressurized fluids are spread all over the Earth. They appear due to geological reasons and cause problems to drillers. Pressurized subsurface reservoirs are difficult to detect. Hence, it is easy to penetrate one when drilling a well. Pressurized fluid then goes into the drilling anulus causing damage to drilling equipment and endangering lives of drillers. The most prominent example is the Deep Water Horizon catastrophe. Wells may be deep. Therefore, it may take up to 20 min for an influx of pressurized fluid to reach the surface. During this time it is not very easy to realize that an influx has happened - at the surface, signs of an influx

*Corresponding author
Email address: `nepomnyi@oreonstate.edu` (Ivan Nepomnyashchikh)

are usually subtle - and to take available preventive measures such as shutting down the well. That is why, drilling industry strives to develop a method of early influx detection.

Motivation: current goal. One of the approaches was developed at DOE NETL (Albany, OR site) [2]. It aims to use geological sensors which are installed at the drilling bit. It is presumed that sensors' data may give an insight to changes in the flow due to influx. Before moving on to the idea's implementation, a valid flow model should be constructed. When a well is being drilled, drilling fluid is pumped into the anulus to wash the cuttings out. Hence, a single-phase flow of drilling fluid is always present in the anulus during a normal drilling operation. When an influx occurs, drilling fluid mixes with the influx fluid and a multiphase flow occurs in the anulus. There are different multiphase flow models. For the stated goal, flow model requiring little computational effort is necessary because calculations should be completed before the influx reaches the surface. The only model which allows for quick computation and preserves sufficient fidelity is drift-flux model (or DFM). Influx simulation with drift-flux model (DFM) is not well covered in the literature (see e.g., [3] and [4]). Therefore, the current goal is to develop a method of influx simulation using drift-flux model.

The developed package. Due to the nature of drift-flux model (DFM), it was decided to simulate influx by adding boundary conditions and equations to the DFM at the spatial coordinate which matches the influx occurrence point in the well bore. This idea was tested on the simplest case of steady influx of air into the steady vertical single-phase flow of water. Package called *stin* [5] was developed for this purpose. "stin" stands for steady influx. The inputs it takes are an influx parameters; the outputs are graphs showing distribution of each of the influx parameters along the well bore.

2. Methodology

2.1. Nomenclature

t - time coordinate

x - spatial coordinate

α_L - liquid phase volume fraction

α_G - gaseous phase volume fraction

ρ_L - liquid phase density

ρ_G - gaseous phase density

ρ_m - mixture velocity

v_L - liquid phase velocity

v_G - gaseous phase velocity

v_m - mixture velocity

f - mixture friction factor

D_h - hydraulic diameter

g - gravitational acceleration

c_G - acoustic velocity in the gaseous phase

γ - gas dependent adiabatic coefficient

C_0 - distribution parameter

v_s - drift velocity

h - numerical spatial step

n - number of a numerical spatial step

2.2. DFM formulation

Drift-flux model (DFM) is devoted to describe 1D gas-liquid flow. General formulation of drift-flux model (DFM) for a vertical flow consists of [6]:

two continuity equations for each of the phases:

$$\frac{\partial(\alpha_L \cdot \rho_L)}{\partial t} + \frac{\partial(\alpha_L \cdot \rho_L \cdot v_L)}{\partial x} = 0$$

$$\frac{\partial(\alpha_G \cdot \rho_G)}{\partial t} + \frac{\partial(\alpha_G \cdot \rho_G \cdot v_G)}{\partial x} = 0$$

Navier-Stokes equation for the mixture:

$$\frac{\partial(\alpha_L \cdot \rho_L \cdot v_L + \alpha_G \cdot \rho_G \cdot v_G)}{\partial t} + \frac{\partial(p + \alpha_L \cdot \rho_L \cdot v_L^2 + \alpha_G \cdot \rho_G \cdot v_G^2)}{\partial x} = -\rho_m \cdot (g + \frac{2 \cdot f \cdot v_m^2}{D_h})$$

This system of equations consists of 3 equations for 6 unknowns:

$$v_L(x) \quad v_G(x)$$

$$\alpha_L(x) \quad \alpha_G(x)$$

$$\rho_G(x) \quad p(x)$$

Therefore, closure relations are required. They are:

$$v_m = \alpha_L \cdot v_L + \alpha_G \cdot v_G$$

$$\rho_m = \alpha_L \cdot \rho_L + \alpha_G \cdot \rho_G$$

$$\alpha_L + \alpha_G = 1$$

$$\rho_L = constant$$

$$c_G^2 = \frac{\gamma \cdot p}{\rho_G}$$

$$v_G = C_0 \cdot v_m + v_s$$

The last equation is called slip relation and represents the key idea of the drift-flux model: phases are allowed to move relatively to each other. The listed closure relations introduce the following parameters:

C_0 - distribution factor; accounts for increase of velocity of the gaseous phase due to gas trend to concentrate in the middle of the flow.

v_s - drift (or slip velocity); accounts for increase of velocity of the gaseous phase due to buoyancy force.

f - friction factor; represents friction force stopping the flow.

γ - gas dependent adiabatic coefficient; used in the equation for gas phase acoustic velocity which, in turn, is used relate gas phase density and flow pressure (assuming that the gas phase acoustic velocity is constant).

2.3. Approach to influx modeling

To start with, the simplest case of steady reduced drift-flux model (DFM) was chosen (reduced means absence of local and convectional accelerations):

$$\frac{\partial(\alpha_L \cdot v_L)}{\partial x} = 0$$

$$\frac{\partial(\alpha_G \cdot \rho_G \cdot v_G)}{\partial x} = 0$$

$$\frac{\partial p}{\partial x} = -\rho_m \cdot (g + \frac{2 \cdot f \cdot v_m^2}{D_h})$$

Consider a single-phase vertical 1D flow of liquid. Assume that at some position along the flow an influx of gas happens. Schematically, it would mean that at some position along the flow a phase has occurred. Hence, we refer to influx as phase occurrence.

As one may assume, phase occurrence can be modeled by setting α_G to zero and then increasing it. This approach doesn't work since not all of the closure relations (see closure relations in the section 2.1) can handle situation when $\alpha_G = 0$.

Therefore, a more mathematically fundamental approach was taken to model phase occurrence. Without providing here detailed mathematical analysis (since, it's not the main purpose of the current report), we state that phase occurrence entails:

1) introduction to the system of equations (DFM) new parameters describing gaseous phase flow (such as velocity, density and volume fraction of the gaseous phase);

2) introduction to the system of equation (DFM) new boundary conditions accounting for new parameters;

3) introduction to the system of equations (DFM) new equations relating new parameters.

It means that there is a point of singularity at the position of phase occurrence, where new boundary conditions, new closure relations, new governing equations appear.

This leads to a conclusion that the entire flow cannot be described continuously with one system of equations. Instead, two systems should be used.

The first one - a regular system of continuity equation and Navier-Stokes equation - describes the single-phase flow from $x = 0$ to $x = L$ (assuming that at $x = L$ an influx has occurred).

The second one - drift-flux model (DFM) describes the two-phase flow starting from $x = L$.

At the positions of the influx, the results of single-phase flow modeling are taken as boundary conditions for drift-flux model (DFM). I.e., the results of single-phase flow modeling at $x = L$ are velocity and pressure of the single-phase flow at $x = L$. They are assumed to be boundary conditions for liquid phase velocity and for mixture pressure of the two-phase flow. Other boundary conditions for the two-phase flow - gas phase velocity, gas phase volume fraction, liquid phase volume fraction, gas phase density - are to be introduced manually.

It is worth mentioning, that the system describing single-phase flow can be solved analytically. Drift-flux model (DFM), in turn, can only be solved numerically. Numerical solution is based on the explicit (forward) Euler's method.

For the sake of better understanding the software package, the solution scheme is provided.

Once again, the flow starts at x = 0 as a single phase liquid flow, defined by boundary conditions for flow velocity and flow pressure and governed by single-phase flow mathematical model (density of the liquid is assumed constant). Two-phase flow starts at x = L, is defined by boundary conditions for velocities of both phases, volume fractions for both phases, gas phase density and pressure of the two-phase flow (i.e., of the mixture) and governed by drift-flux model (DFM).

First, the single-phase flow model is solved ($x = L$ is the position of influx occurrence, $x = 0$ is where the flow starts.):

For $0 \leqslant x \leqslant L$:

$$v(L) = v(0)$$

$$p(L) = -\rho_L \cdot (g + \frac{2 \cdot f \cdot v(0)^2}{D_h}) \cdot L + p(0)$$

Then, the drift-flux model is solved (n is the number of a spatial step, h is the magnitude of the spatial step); calculations must be done in the order the equations appear:

For $x \geqslant L$:

$$p_{n+1} = p_n + h \cdot [-(\alpha_{L_n} \cdot \rho_L + \alpha_{G_n} \cdot \rho_{G_n}) \cdot (g + \frac{2 \cdot f}{D_h} \cdot (\alpha_{L_n} \cdot v_{L_n} + \alpha_{G_n} \cdot v_{G_n})^2)]$$

$$\rho_{G_{n+1}} = \frac{\gamma}{c_G^2} \cdot p_{n+1}$$

$$v_{G_{n+1}} = v_{G_n} + h \cdot [-\frac{\gamma}{c_G^2} \cdot \frac{v_{G_n} \cdot \alpha_{G_n} \cdot C_0}{\rho_{G_n}} \cdot \frac{\mathrm{d}p}{\mathrm{d}x}|_n]$$

$$\alpha_{G_{n+1}} = \alpha_{G_n} + h \cdot [-(\frac{\gamma}{c_G^2} \cdot \frac{\alpha_{G_n}}{\rho_{G_n}} \cdot \frac{\mathrm{d}p}{\mathrm{d}x}|_n + \frac{\alpha_{G_n}}{v_{G_n}} \cdot \frac{\mathrm{d}v_G}{\mathrm{d}x}|_n)]$$

$$\alpha_{L_{n+1}} = 1 - \alpha_{G_{n+1}}$$

$$v_{L_{n+1}} = v_{L_n} + h \cdot (\frac{v_{L_n}}{\alpha_{L_n}} \cdot \frac{\mathrm{d}\alpha_G}{\mathrm{d}x}|_n)$$

3. Implementation

3.1. Description of the package's structure

Software package is called stin [5], which stands for steady influx. It is written in Python 3.7 [7] and can be installed as a python package. It can't be installed as a separate piece of software.

Software is designed to program the solution scheme presented at the end of section 2.3. The concept of package usage assumes that a user inputs from command line initial volume fraction of the influx, other boundary conditions are to be input from the input file, the output is 6 plots representing distribution of each parameter of the two-phase flow versus spatial coordinate.

The structure of the package is presented below.

- stin
    - docs

- stin
  - __init__.py
  - __main__.py
  - boundary_conditions.py
  - functions.py
  - parameters.py
  - solution.py
  - inpyt.yaml
  - tests
    - __init__.py
    - test_unit.py
    - test_integration.py
- _version.py
- .gitignore
- .travis.yaml
- CHANGELOG.md
- LICENCE
- MANIFEST.in
- README.md
- Requirements.txt
- setup.cfg
- setup.py

Implementation of the solution scheme is contained withing the module *stin*. It is also the main module of the package.

File *functions.py* contains python functions for all the equations in the solution scheme. For the single-phase model: both of the equations are presented as python functions. For the two-phase model: only the terms on the right-hand side of each of the equations are presented as python functions.

File *solution.py* contains two functions. The first one runs the solution scheme in a for loop, employing python functions from the *functions.py* file. As an output it provides python lists of each of the two-phase flow variable. As an input it takes the value of initial influx volume fraction. Within its body it contains a procedure of obtaining other boundary conditions from the *input.yaml*, *parameters.py* and *boundary_conditions.py* files. The second one takes the output of the first function and plots the lists versus spatial coordinate. The output of this function is the output of the entire package.

File *__main__.py* contains procedure which allows to input initial influx volume fraction from command line. It calls both functions from the *solution.py* file. It calls the plotting function withing the procedure of multiprocessor calculations. Thus, results' plotting is realized using python multiprocessing.

File *input.yaml* is an input file. It contains boundary conditions for the single-phase flow model as well as parameters necessary for spatial step calculations.

File *parameters.py* contains parameters of the drift-flux model (DFM). DFM parameters are a matter of another research work, they can't be modified by user and, thus, put into a file not accessible for a user.

File *boundary_conditions.py* defines an influx class. An influx is defined by its initial values which are the boundary conditions for the two-phase flow model. Hence, basically this class defines boundary conditions for the drift-flux model (DFM). User is only allowed to assign half of the boundary conditions for the two-phase flow model. The second half is defined withing the boundary_conditions class in order to verify that all the boundary conditions comply with the equations of the two-phase flow model.

Tests module contains units test for all the functions in the *functions.py* file and integration tests for the solution function from the *solution.py* file. It doesn't contain integration tests for the plotting function. Internal, edge and corner cases are tested for all the appropriate functions in *test_unit.py*

All the other files and modules in the package are not related to the programming of the solution and necessary for packaging and distribution of the software.

Docs module contains *Sphinx* [8] files necessary for the documentation website.

_version.py file contains the version of the package and is used in other files when reference to the version is necessary. Version is done in accordance with the requirements of semantic versioning [9].

.travis.yml file is the input file for Travis CI [10].

requirements.txt file dictates Travis CI [10] what dependencies are needed to be installed.

CHANGELOG.md file tracks the changes in each version release according to the keep changelog website [11].

LICENCE file contains the MIT licence to determine the rules of the package usage.

MANIFEST.in file is mainly necessary to include *input.yaml* file into the installation process since not code file containing within a module can be removed to other directories during installation.

README.md file contains all necessary information on the package installation and getting started with.

setup files determine installation process.

3.2. Dependencies

The package is dependent on the following third party python libraries:

- matplolib [12]
- pyyaml [13]

Testing is done with the help of:

- pytest [14]
- pytes-cov [15]

Initially, code was optimized through introduction of the numpy library. But then it was removed due to the difficulties associated with its integration to the block of the code responsible for multiprocessing. Still, using numpy library is one of the future goals as it helps with code optimization.

3.3. Running the package

stin [5] is a python package. It doesn't contain an executable file and can't be installed directly on the operational system. It employs python tools for installation and deinstallation. Hence, one should have python installed on the computer in order to install the package.

It is not a python library. It is merely a piece of software written in python and using python tools for installation.

The major requirement for the user is to have the version of python greater than or equal to 3.7 [7]. The package uses importlib.resources which became a part of python standard library since the release of version 3.7.0.

The package can be downloaded either from source or from TestPyPI. For detailed guidance on installation process, see the package's documentation website.

Installation from source. The package can be downloaded from stin. Then type in the command line: *pip install <path to the package's root directory>*. For deinstallation, type in the command line: *pip uninstall stin.*

Installation from TestPyPI. The package can be installed directly from TestPyPI. For that purpose, type in the command line: *pip -i https://test.pypi.org/simple/ stin.* For deinstallation use: *pip uninstall stin.*

Running the package. After the package has been installed, one can run it using command line interface. The package does not have GUI.

In order to run the package, type in the command line: *py -m stin -alpha 0.01* or *py -m stin –initial_gas_fraction 0.01.*

One may notice that user should specify a variable when running the package. This variable is initial influx volume fraction or, in other words, gas phase volume fraction boundary condition for the two-phase flow model.

It is important to bare in mind that for the current version of the package only small values of initial influx volume fraction are recommended to be used. Thus, in the preceded example initial influx volume fraction equals to 0.01. The range of possible initial influx volume fractions is from 0 to 1, not including the margins. Currently, initial influx volume fraction values not bigger than 0.4 are recommended.

Command line interface contains the *help* command also. It can be called by typing in command line: *py -m stin -h* or *py -m stin –help*. In the last case, put two dashes before the *help*.

4. Results

As a result of running the package, a user gets 6 plots representing distribution of each of the parameters of the two-phase flow versus spatial coordinate.

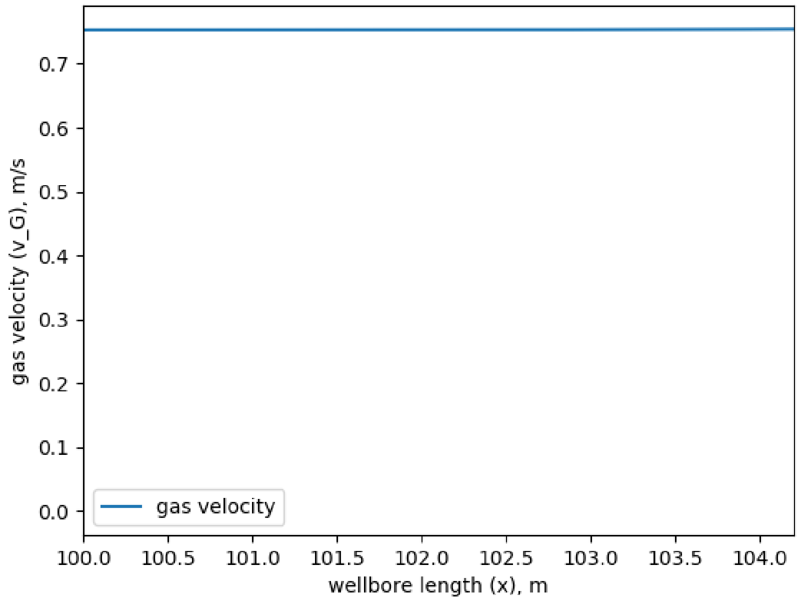

Figure 1: Gaseous phase velocity as a function of spatial coordinate [16]

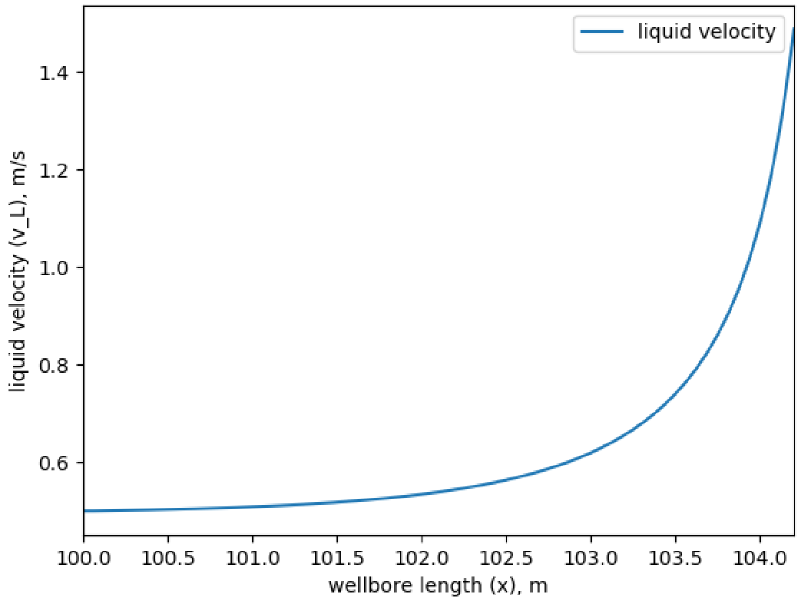

Figure 2: Liquid phase velocity as a function of spatial coordinate [16]

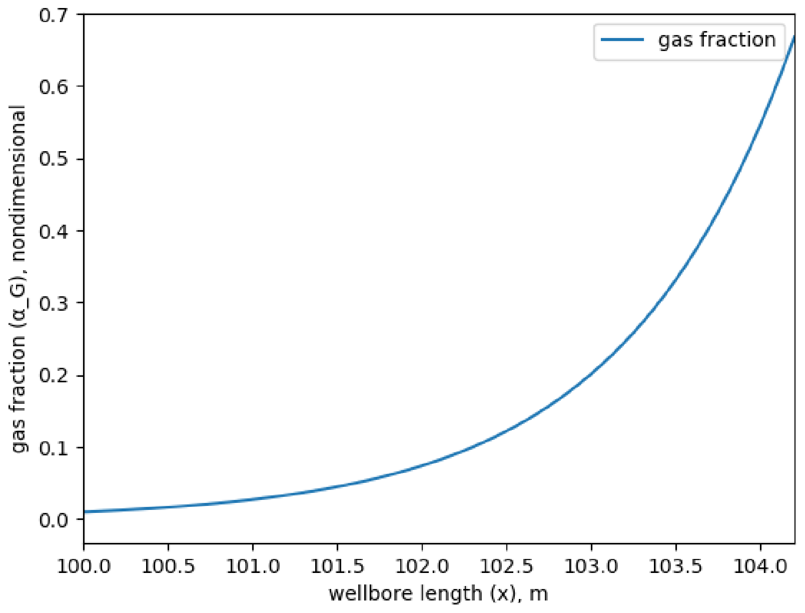

Figure 3: Gaseous phase volume fraction as a function of spatial coordinate [16]

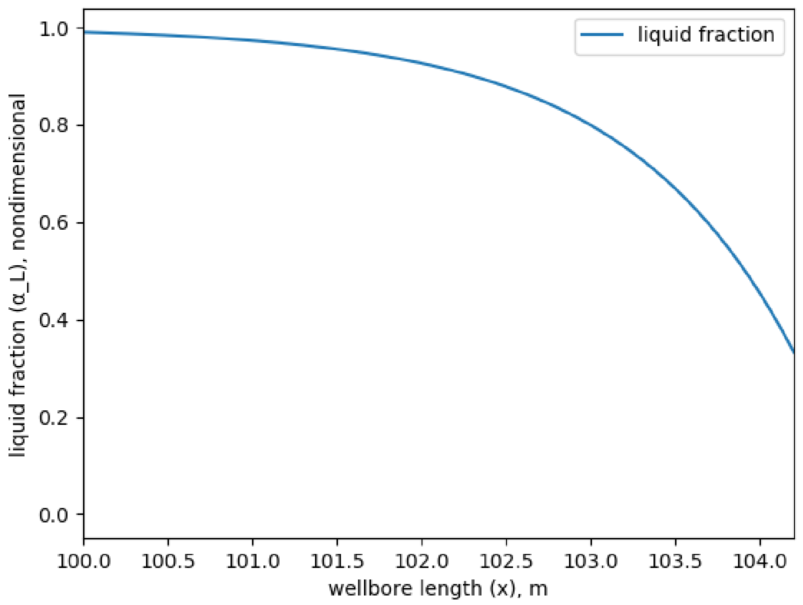

Figure 4: Liquid phase volume fraction as a function of spatial coordinate [16]

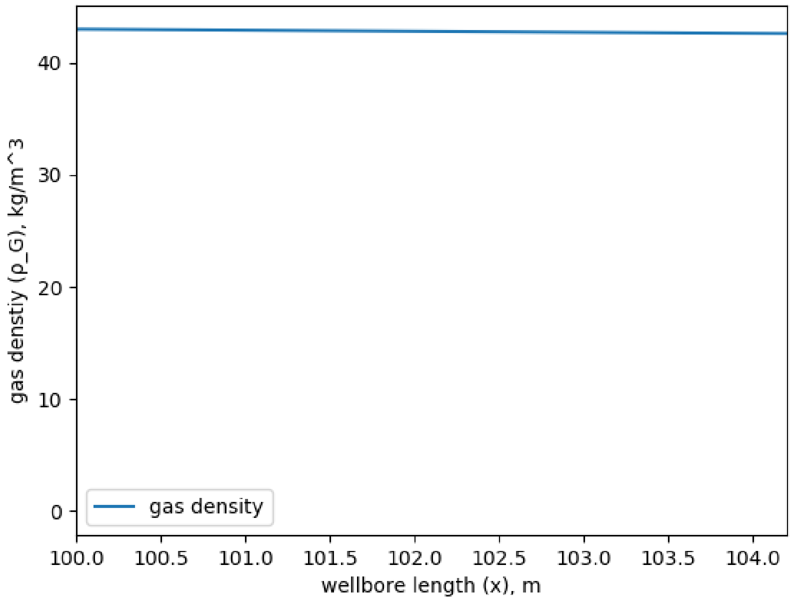

Figure 5: Gaseous phase density as a function of spatial coordinate [16]

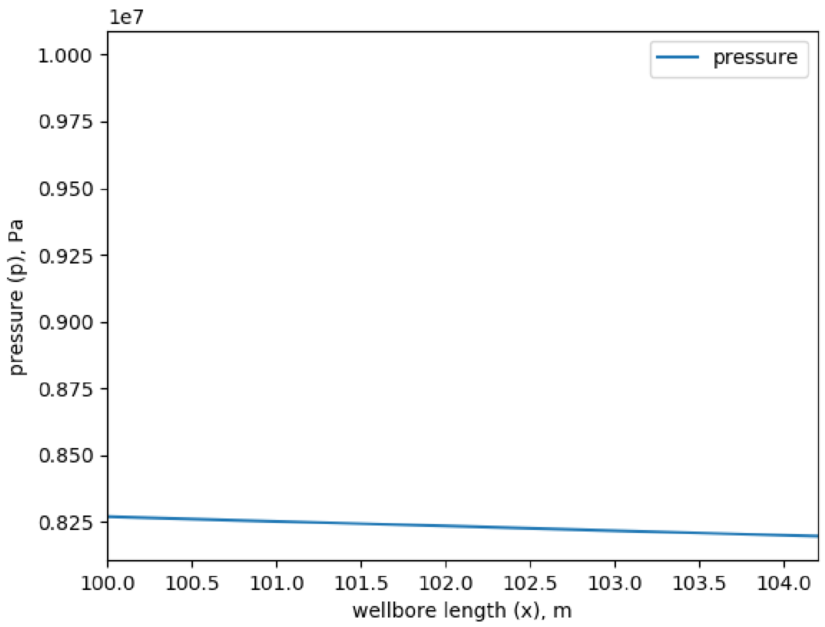

Figure 6: Two-phase flow pressure as a function of spatial coordinate [16]

The shown results are valid for initial single-phase flow velocity $0.5 m/s$, initial single-phase flow pressure $10 \cdot 10^6 Pa$, initial influx volume fraction 0.01.

The graphs are plotted only for the two-phase portion of the flow. Influx occurs at $100m$ from the bottom of the vertical pipe. That is why, the beginning values of spatial coordinate on all the plots is $100m$.

The extreme right value of spatial coordinate doesn't show the position where the influx vanishes. Instead, horizontal axis is limited by the range of validity of the model. I.e., at the extreme right value of spatial coordinate, the conditions imposed on drift-flux model ceased been satisfied. These conditions are not the topic of the current report and, therefore, haven't been discussed here.

Better scaling of the plots is a part of future work. As well as representation of the entire flow on the plots (i.e., from $x = 0$ to the end of the validity range).

5. Conclusions

The developed software simulates two-phase portion of the vertical flow of incompressible liquid which has been experiencing steady influx of gas (air) at some position.

The developed software is based on drift-flux model (DFM).

The results show that the proposed approach to simulate influx using DFM is potentially correct. Overall comparatively to the previously used approaches to influx modeling, the one employed in the current package is mathematically correct and computationally faster.

Analysis of mathematical behavior of the model done by the author using the software revealed valuable insights on approach to develop expressions for drift-flux model (DFM) parameters C_0 and v_s. This will be used in the future to augment the model with correct equations for drift-flux model (DFM) parameters in order to achieve correct performance of the software for a wide range of flow conditions. Which is the ultimate goal for this package.

Moreover, the region of transition from single-phase flow to two-phase flow will be better explored and augmented with necessary equations if needed.

Alpha release of the software is anticipated in spring 2020. It will have proper expressions for drift-flux model parameters C_0 and v_s as well as be able to handle transient gas-liquid flows.

Beta release of the software is anticipated in autumn 2020 with the subsequent release of v1.0 in spring 2021.

After that, work on modeling liquid-liquid flows will begin. Version 2.0 is devoted for handling liquid-liquid flows.

The software can be used both in industry and academia when fast simulation of two-phase flows without high fidelity is necessary.

As it has already been mentioned previously, the global goal it to be able to predict model's boundary conditions based on the parameters at some point along the spatial coordinate. The code used in the package will became a part of the code devoted for achievement this global goal.

Appendix. Data usage

Software described in the current report is available openly via the *stin* software package [5]. The most recent version of *stin* can be found at its GitHub repository. All figures and plotting scripts necessary to reproduce them are available openly under the CC-BY license [16].

References

[1] K. Niemeyer, Software development for engineering research syllabus (2019).
URL `https://github.com/SoftwareDevEngResearch/syllabus-s2019`

[2] B. Tost, et al., Kick detection at the bit: Early detection via low cost monitoring, NETL-TRS-2-2016; EPAct Technical Report Series; U.S. Department of Energy, National Energy Technology Laboratory: Albany, OR, 2016; p. 48.

[3] U. J. F. Aarsnes, et al., Control-oriented drift-flux modeling of single and two-phase flow for drilling, Proceedings of the ASME 2014 Dynamic Systems and Control Conference DSCS2014, October 22-24, 2014, San Antonio, TX, USA.

[4] A. Nikoofard, et al., State and parameter estimation of a drif-flux model for under-balanced drilling operations (2017).
URL `https://doi.org/10.6084/m9.figshare.8265695.v1`

[5] I. Nepomnyashchikh, stin v0.2.1 (2019). `doi:10.5281/zenodo.3243702`.
URL `https://github.com/SoftwareDevEngResearch/stin`

[6] A. Ambrus, et al., Real-time estimation of reservoir influx rate and pore pressure using a simplified transient two-phase flow model, Journal of Natural Gas Science and Engineering, 2016.
URL `http://dx.doi.org/10.1016/j.jngse.2016.04.036`

[7] G. van Rossum, Python v3.7.3 (2019).
URL `https://www.python.org/`

[8] G. Brandl, et al., Sphinx: Python documentation generator (2019).
URL `http://www.sphinx-doc.org/en/master/index.html`

[9] T. Preston-Werner, Semantic versioning 2.0.0 (2019).
URL `https://semver.org/`

[10] P. Sarnacki, et al., Travis ci: continuous integration service (2019).
URL `https://travis-ci.com/`

[11] O. Lacan, keep a changelog: Don't let your friends dump git logs into changelogs, version 1.0.0 (2019).
URL `https://keepachangelog.com/en/1.0.0/`

[12] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (3) (2007) 90–95. `doi:10.1109/MCSE.2007.55`.

[13] K. Simonov, Pyyaml v5.1.1 (2019).
URL https://github.com/yaml/pyyaml

[14] H. Krekel, et al., pytest v4.6.3 (2019).
URL https://github.com/pytest-dev/pytest

[15] M. Schlaich, pytest-cov v2.7.1 (2019).
URL https://github.com/pytest-dev/pytest-cov

[16] I. Nepomnyashchikh, Reproducibility package for the report on stin v0.2.1 (2019). doi:10.6084/m9.figshare.8265695.v1.
URL https://doi.org/10.6084/m9.figshare.8265695.v1