iFlow Getting Started Tutorial

Yoeri Dijkstra[1], Ronald Brouwer[1, 2], Henk Schuttelaars[1], and George Schramkowski[1, 2]

[1]Delft University of Technology, Department of Applied Mathematics. Delft, Netherlands
[2]Flanders Hydraulics Research. Antwerp, Belgium

December 28, 2017

This user guide provides a short and practical introduction to iFlow using several example input files. You will learn how to run the model, change the input and plot results. We also give a short introduction to the iFlow data structure and framework so that you can start making custom plots and modules.

We strongly recommend reading the iFlow paper Dijkstra et al. (2017) (freely accessible at `https://doi.org/10.5194/gmd-10-2691-2017`), which discusses the philosophy and framework of iFlow as well as the equations that are solved and the technique used to do this. These subjects are not treated in detail in this tutorial. This tutorial and the scientific paper can be read in any order.

in Section 1 of this guide you will learn how to run your first simulation. Section 2 will explain the structure of the input file and you will learn how to adjust input parameters and model geometry. Section 3 shows one powerful tool of iFlow: the ability to do quick sensitivity studies. Next, Section 4 will present some options for plotting model results using iFlow's simple tools for plotting (`STeP`). Up to this point, this tutorial can be done easily by users that are new to programming. Sections 5 and 6 require some python programming. Section 5 discusses more customised plotting tools, focussing on the way you can use the data generated by the model. Finally, Section 6 gives a short introduction into programming extensions to iFlow.

Part I
Getting started

1 Introduction

1.1 For those new to Python or programming

iFlow is written in Python 2.7, but running the model does not require any knowledge of programming. Users may however soon feel the need for customised plots, for which we recommend learning how to work with Python's *matplotlib* plotting toolbox. Alternatively, iFlow provides a feature to export simulation data to `.mat` files for plotting in Matlab.

iFlow provides a friendly environment for new programmers to extend the model. iFlow has a modular structure, which means that any extension to the model may be started in a new Python script that needs to satisfy some basic rules (see Section 6). Therefore you will not need to dive into the existing code, but just add your own using your own programming style.

1.2 Installation

iFlow requires Python 2.7 and several common python packages, including numpy, scipy and matplotlib. We strongly advise installing a pre-built package such as Anaconda or Canopy. Both packages are free and available on all popular platforms. When downloading Anaconda or Canopy, make sure you choose the Python 2.7.... option.

Download the iFlow source code from GitHub (direct zip link `https://github.com/YoeriDijkstra/iFlow/zipball/master`[1]). Unpack this folder to a local disk or network location where you have reading and writing rights. All the iFlow source code is located in the iFlow folder.

Next, download the iFlow tutorial package from GitHub (direct zip link `https://github.com/YoeriDijkstra/iFlowTutorial/zipball/master`[2]) and unpack this folder to a local disk or network location where you have reading and writing rights. This folder contains this tutorial as well as example iFlow files.

iFlow can be run from the command line or using any IDE (i.e. graphical code editing and running software) that supports Python. We recommend using an IDE. Spyder is a light-weight IDE for Python that is a popular default choice and can be used for running iFlow. We ourselves use PyCharm. Any other IDE should work as well. Both aforementioned IDEs are free software.

1.3 Running your first simulation

If running iFlow from the command line[3], navigate to the iFlow directory by typing `cd` followed by the path to the iFlow main directory and press enter. If this is on a different disk than the default location e.g. D:, type `D:` and press enter to move to the disk first and then navigate to the iFlow directory. Type `python main.py` to run iFlow.

If running iFlow using an IDE, import/load iFlow into the IDE and run the file 'main.py' in the iFlow folder.

The iFlow main menu should appear on the screen. The main menu should look like the text below (version number may be different)

```
iFlow version 2.4.3

No working directory set. Now working from the iFlow root directory.
Enter cwd to change the working directory.

Please enter the path to a new input file:
```

iFlow works using a main directory for its source files and a working directory for specific projects. The working directory is used for input and output files and own pieces of code. By default, the working directory is the same as the main directory. However, we strongly

[1]link to git repository https://github.com/YoeriDijkstra/iFlow.git
[2]link to git repository https://github.com/YoeriDijkstra/iFlowTutorial.git
[3]On a Windows system, in the start menu, type `cmd` and press enter to start the command line

advise to work from a separate working directory for a clean work flow. Here we use the folder 'iFlowTutorial' as working directory. To change the working directory type `cwd` in the menu and press enter. Next type (or copy) the full absolute path to the tutorial folder and press enter. Note that the menu re-appears with the line

```
Current working directory set to [...]/iFlowTutorial.
```

To run a simulation, enter the path to an iFlow input file. Type the path relative to working directory, including the file name and extension. For this tutorial, type

```
input/inputEMS.txt
```

iFlow then runs the input file and shows its progress on the screen. This should look like the text below.

```
Call stack was built successfully
** Call stack **
1 general.Geometry2DV
2 numerical2DV.RegularGrid
3 numerical2DV.turbulence.Uniform
4 numerical2DV.hydro.HydroLead tide, river
5 numerical2DV.hydro.HydroFirst tide, stokes, river, nostress, adv
6 plot.Plot_base_model

** Run **
Running turbulence model Uniform
Running module HydroLead
Running module HydroFirst
```

Also, some figures showing model output should appear on your screen. Close these for now, we will look at them later.

Let us take a closer look at the console output above. Under `** Call stack **` iFlow lists the modules that it will use in the simulation. Modules are separate components of the model each with their specific task. Each module name consists of a package name and the module name separated by dots. The above model will make the geometry of the estuary, build a grid, determine the turbulence related parameters and then compute the hydrodynamics at leading and first order. The last module shows plots of the result. Under `** Run **` some of the modules show that they are running or show their progress otherwise.

Now restart the model. The menu will appear again, but now shows

```
Please choose an input file from the list of recent files:
1 input/inputEMS.txt
or enter the path to a new input file:
```

iFlow saves the paths to the five most-recently used input files. To run them, type the number in front of the input file and press enter. Here, press `1` and press enter. To run the last-used input file, you can alternatively omit the number `1` and just press enter.

2 Understanding and changing the input file

Now we have run our first simulation, let us have a closer look at the input file. Go to the `iFlowTutorial` folder and then to `input` and open the file `inputEms.txt` in the IDE or any editor[4].

The lines starting with `#` are comments and are not interpreted. The input file contains several blocks starting with the keyword `module`. This keyword is followed by one or more references to modules. These references consist of two parts: the main package (=folder) that the module is in and the module name, separated by a dot. The default set of packages is located in the folder `packages` in the iFlow main directory (e.g. `general` or `numerical2DV`). Custom packages may be located in the current working directory of modules. In this example `plotting` is such a custom package.

The input variables and their values are listed in each module block. The variables directly under a module reference belong to the that module. To learn what variables are required for each module, we refer to the iFlow manual. However, more practically, this tutorial discusses many modules in iFlow 2.4 and we advise starting any new input file by using the tutorial files as a basis. The Ems input file is shown below.

```
# Input file
#
# Date: 13-10-2017
# Authors: Y.M. Dijkstra, R.L. Brouwer

## Geometry ##
module  analytical2DV.Geometry2DV
L       64.e3
B0      type    functions.Exponential
        C0      670
        Lc      30.e3
H0      type    functions.Polynomial
        C       -1.9e-18 2.1e-13 -4.8e-09 -1.5e-04 1.1e+01

## Method ##
module     numerical2DV.RegularGrid
xgrid      equidistant     100
zgrid      equidistant     50
fgrid      integer         2

xoutputgrid    equidistant     100
zoutputgrid    equidistant     50
foutputgrid    integer         2

## Hydrodynamics ##
module      semi_analytical2DV.HydroLead semi_analytical2DV.HydroFirst
submodules  tide river adv baroc stokes nostress
```

[4]For those new to programming: never use Word or similar text editors. Simple tools such as Notepad will do for now, but soon you may want to install a dedicated code editor or IDE.

```
A0          0 1.39 0
A1          0 0 0.22
phase0      0 0 0
phase1      0 0 -173
Q0          0
Q1          65

## Turbulence ##
module      analytical2DV.TurbulenceUniform
Av0amp      0.019
Av0phase    0.
sf0         0.04
m           0.
n           0.

module      numerical2DV.DiffusivityUndamped
sigma_rho   1.

## Salinity ##
module       analytical2DV.SaltHyperbolicTangent
ssea         30
xc           -1.e3
xl           11.88e3

## Sediment ##
module       semi_analytical2DV.SedDynamic
submodules   erosion noflux sedadv
astar        1.e-5
ws0          0.5e-3
Kh           100

## Plotting ##
module          plotting.Plot

## Output ##
module       general.Output
path         output
filename     testOutput

requirements    zeta0 zeta1 u0 u1 T F a c0 c1 c2
```

2.1 Geometry

The standard geometry package is `analytical2DV.Geometry`. The package sets the length `L`, width `B` and bed level `H`. The width and bed level may be set by any one of several functions that should be entered behind the keyword `type`. The standard functions are in the `functions` package. Some of the most useful functions are

- Constant. Requires one parameter `C0`
- Linear. Requires the level at $x = 0$ `C0` and the level at $x = L$ `CL`
- Exponential. An exponential profile of the form $C0e^{-x/Lc}$, requires `C0` and `Lc`
- Polynomial. A polynomial with coefficients `C`. The number of coefficients sets the order of the polynomial
- PolynomialLinear. A polynomial with coefficients `C` up to coordinate $x =$`XL` and linear from this point in such a way that the function is continuous and once differentiable

Assignment:
Open the Ems input file and change the width of the estuary using any of the above functions. How do the plotted results change? What happens if the width becomes negative at any point?

2.2 Grid

Under the module `RegularGrid` you can define the grid resolution for the computation and of the output. It may help to save disk space if the output resolution is lower than the computation resolution. The grid has three dimensions: x, z and f, which represent the along-channel, vertical and frequency dimension. Behind each of the dimensions is the definition of the type of grid axis (here we will only use `equidistant`) and the number of grid cells. The iFlow grid has points at the boundaries and therefore the number of grid points is the number of grid cells +1. Here we will only work with the case of three frequency components; i.e. two 'cells' in the frequency dimension. This represents the subtidal, M_2 and M_4 signal.

2.3 Hydrodynamics

The hydrodynamics section has two modules: `HydroLead` and `HydroFirst`, computing the first and leading order hydrodynamics (see Dijkstra et al. (2017)). The variables contain the keyword `submodules`, which lists the flow components computed by this module. The other input variables contain a number to indicate to which order the belong (0: leading order, 1: first order). `A` indicates the tidal amplitude at the mouth followed by number according to the frequencies subtidal, M_2 and M_4. `phase` indicates the phase (in deg) of the tidal amplitude at the mouth also followed by the frequency component. Finally, `Q` is the river discharge, entered as a positive number for downriver flow.

2.4 Salinity

The salinity model in this tutorial is a simple hyperbolic tangent profile $s_{\text{sea}}/2\left(1 - \tanh\left(\frac{x-x_c}{x_l}\right)\right)$. This module requires parameters `ssea`, `xc` and `xl`.

2.5 Sediment dynamics

The semi-analytical sediment module has three submodules (see Dijkstra et al. (2017)) and requires three variables: `astar`, a measure for the total amount of sediment in the system, `Kh`, the horizontal dispersion coefficient and `ws`, the settling velocity. The numerical sediment module allows some more flexibility, e.g. by allowing one to prescribe the subtidal sediment concentration on the seaward boundary instead of the total amount of sediment in the system. This feature will also be included in the semi-analytical module in the next iFlow version.

2.6 Output

The output module requires variables `path` and `filename` for writing the output file. The output is written as binary files using the Python module Pickle and we will explore how to read these files in Section ??.

The last line of the input file starts with the keyword `requirements`. This line tells iFlow which variables you want to compute. iFlow will omit any modules that are mentioned in the input file, but which' output is not necessary to compute these output variables. iFlow will also give you a warning if not all variables can be computed, so this can also be used to check if you have supplied all the correct modules. This `requirements` line may be used directly following the ouput module, but may also be used without the output module in case you do not want to save any result, but just test it.

3 Sensitivity analysis

One of the strengths of iFlow is the ability to do fast sensitivity analyses using the module `general.Sensitivity`. This module allows fully automatic sensitivity analyses over any number and any selection of parameters. In the Ems input file you will find a block for the sensitivity analysis module, which is commented out by default. Uncomment it (remove `#`) and this will look like

```
## Sensitivity
module       general.Sensitivity
variables    Q1 ws
Q1           np.linspace(20, 80, 10)
ws 0         10**np.linspace(-4, -2, 10)
loopstyle    permutations
```

Behind `variables` you can list any number of parameters that are input to the model. Here we use two variables: the river discharge and settling velocity. These variables each have their own line to specify the range of values to test. The model will try to interpret this range as python code, where `np` may be used to refer to `numpy` code[5]. The final variable is `loopstyle`. This may have the value `permutations` to indicate that all combinations of variables should be tested, or `simultaneous` to indicate that all variables should be varied at the same time.

Additionally, uncomment the line

```
iteratesWith    general.Sensitivity
```

under the output module. This tells the ouput to write a new file with every iteration of the sensitivity analysis. It is possible to give meaningful names to the output files by using dynamic naming. For example

[5]The numpy `linspace` command behind `Q1` for example tell the model to run 10 values from 20 to 80 (including endpoints) with equal distance between them. The command behind `ws` tells to take the power of 10 of 10 values between -4 and -2.

```
path output/Ems_sensitivity
filename  out_Q@{'Q1'}_ws@{'ws0'}
```

under the output module. The `@'...'` indicates that this should be replaced with the value of the variable between quotation marks (' ').

Assignment:
Uncomment the sensitivity module, iteratesWith line and change the output file name to a dynamic name and run the model. Note in the first lines of the console output that iFlow places the sensitivity module after the geometry, grid, turbulence and leading-order hydrodynamics. Why is this possible? See in the console output that iFlow notifies you of the progress. Find the folder with the output files and check if the dynamic naming works.

The results of the sensitivity study for the location of the ETM can be plotted by running the input file `input/readsensitivity.txt`. This runs a plotting module `plotting.ws_Q_plot.txt` and produces one figure with the ETM location (colors) plotted against the settling velocity and river discharge. If the input file does not work immediately, please check if the `folder` in the input file points to the folder with the results of the sensitivity test and that this folder only contains results of the sensitivity test.

4 Quick plotting tools

iFlow offers the Simple Tools for Plotting (STeP) package, which provides tools for quickly making standard plots and ensures a consistent plot style. Plotting is done by special plot modules. An example of a plot module is provided in the tutorial folder. Please find the subfolder `plot` and the file `Plot.py`. Open this using your code editor. You can use iFlow and Python to make any type of plot, but iFlow provides some quick visualisation tools that allow you to plot variables with a single line of code. These tools are:

- `lineplot` - plot a variable of choice along one grid axis.
- `contourplot` - plot a colorplot of a variable of choice along two grid axes.
- `transportplot` - plot the contributions to the sediment transport.

A full explanation on how to use these plots is provided in the comments in the plot module.

5 Exporting data to Matlab

If you want to do post-processing in Python, skip this section and proceed to the next part.

For programmers that want to do post-processing in Matlab, iFlow provides a module to write model results to `.mat` files. In the Ems input file, change `general.Output` to `general.OutputMat`. Run the module (for now put comment signs in front of the sensitivity analysis module to prevent too much output; the output module works just as well for sensitivity analyses). Load the resulting `.mat` file into Matlab. This loads all input and output variables from iFlow into Matlab.

A variable like `u1` (first-order horizontal velocity) appears multiple times followed by a subscript and then the name of underlying contributions to `u1`. Sum all these contributions to get the total first-order horizontal velocity. An explanation of all variables is given in the first pages of the manuals (see Section 9 for more explanation on how to use the manuals)

Part II

Measurements and calibration

6 Another example input file

In this part we will work with the Scheldt River example. It is possible to run this model using the same modules as the Ems River example, but for illustration purposes we have included a few different modules. We have changed the turbulence model to `KEFittedLead`. This is a model where the bed friction scales quadratically with the leading-order velocity and eddy viscosity scales linearly with the depth-averaged leading-order velocity. The model is calibrated using one roughness parameter z_0^*, which is a dimensionless roughness height, $z_0^* = \frac{z_0}{H}$, where z_0 is the roughness height and H is the depth. Note that the roughness height in iFlow is not the same as in complex models that use the $k - \epsilon$ model or similar models.

7 Including measurements

In this section we introduce a way to compare and calibrate model results against measurements. An example module is given in the package (=folder) `measurements` in the iFlowTutorial directory. Open the file `Scheldt.py`. We will go through the `run` method and explore where to make changes to adapt this to other estuaries. The first lines in the run method create a dictionary `d` and a variable `Scheldt_measurements`. For another estuary, change this variable name to any other name. Do this in the entire file.

Next we define station and section names and locations. These are optional and can be replaced by an empty list `[]`. Next specify the length in the variable `L`. This is followed by the water level measurements. iFlow works using a set of tidal components, hence the measurements are entered as M_2 and M_4 components. Enter the location of the water level stations and the amplitudes and phases of the water level components. If some component is not available, fill the array with `np.nan`. If the amplitude is known, but the phase is unknown, fill the phase with zeros. Similarly flow velocity or sediment concentration data may be entered. An example of the velocity is provided in the file. This is optional and may be replaced by `np.nan`.

When changing the variable `Scheldt_measurements`, change this in the entire file and change this in the file `registry.reg` in the folder `measurements`. When changing the name of the script `Scheldt.py`, also change the name of the class behind `class` in the script.

8 Calibration

iFlow provides a simple calibration module for one-parameter calibrations. The calibration module calibrates the M_2 water level to observation data. The example input for the measurements and calibration is provided below.

```
## Measurements ##
module          measurements_iflow.Scheldt


## Calibration
```

```
module                  general.Calibration
measurementset          Scheldt_measurements
calibrationParameter    z0*
z0*                     0.001
changeFactor            1.2
#ignorePhase             True
```

The variable `measurementset` corresponds to the measurement variable in the module (see section above). We notify iFlow that the calibration parameter is `z0*` and provide a starting value for this of 0.0015. The calibration module then tries values that are a factor `changeFactor` (i.e. 1.2) higher and lower than the initial value. From this, it proceeds looking for higher or lower values until it finds a minimum least-squares difference between the modelled and measured M_2 tide. The optional argument `ignorePhase`, if set to True, only calibrates the M_2 water level amplitude and ignores the phase.

Assignment:
Run the module and establish that it converges for a value of z_0^* around 0.00132. Try a few different starting points and change factors (all higher than 1) and see if it converges to the same value and how many iterations it takes. Note that the calibration module is said to have converged if it is within 10% of the solution. (You may change the convergence tolerance in the variable `TOLLERANCE` at the top of the module calibration). If you are done, replace the value of z0 under the turbulence module with the calibrated value and comment-out the calibration module.*

Part III
Manuals and other information

9 Guide to the manuals

iFlow comes with a set of in-depth manuals, located in the `Manuals` folder in the iFlow source directory. The `iFlow modelling framework manual` provides all general information on the framework and how to program your own iFlow modules. It contains an extensive description on the requirements for new modules, the iFlow data structure, iFlow toolboxes and options for complex interdependent modules. Reading this manual is recommended before starting to program your own iFlow modules.

The other modules are named after the packages in the `packages` folder in the iFlow source directory and describe the modules in these directories. Each manual starts with a table describing all the variables that are input and output to these modules and may be used as reference. The rest of the manuals give extensive descriptions of the mathematical model, rather than the programming. It describes the assumptions, derivation and solution method of the mathematical model used in each module. For the hydrodynamics and sediment dynamics, shorter and more abstract descriptions of the model is given in **?**Dijkstra et al. (2017). For more in-depth information, the manuals on the semi-analytical and numerical packages are good and complete introductions to using perturbation methods for modelling hydrodynamics and sediment dynamics in estuaries.

10 License

iFlow is open source software and available through GitHub. When using iFlow in any paper or report, please include a reference to Dijkstra et al. (2017). The source code is available on an LGPL licence. This means that the code is free for commercial and non-commercial use. Although not required, we encourage you to make your own developed modules, scripts and applications open source as well.

References

Dijkstra, Y. M., Brouwer, R. L., Schuttelaars, H. M., and Schramkowski, G. P. (2017). The iFlow Modelling Framework v2.4. A modular idealized process-based model for flow and transport in estuaries. *Geoscientific Model Development*, 10:2691–2713.