Notes on *E. coli* model and code

Tomás Aquino

March 29, 2021

Abstract

These are notes on how to compile and use the code (on a UNIX system) and what processes are implemented.

1 Transport

Downstream transport is assumed to occur only in the water column and by advection only. Spatial variability is neglected. Advection is approximated as piecewise-constant in time. In the code, it is characterized by onset times, corresponding velocities, and an indicator of surge events. For example, the object

$$\{(t_0, v_0, s_0), (t_1, v_1, s_1), (t_2, v_2, s_2)\} \tag{1}$$

represents the flow

$$v(t) = \begin{cases} v_0, & t \in [t_0, t_1[\\ v_1, & t \in [t_1, t_2[\\ v_2, & t \geqslant t_2 \end{cases}. \tag{2}$$

with the surge information

$$s(t) = \begin{cases} s_0, & t \in [t_0, t_1[\\ s_1, & t \in [t_1, t_2[\\ s_2, & t \geqslant t_2 \end{cases}. \tag{3}$$

Surge $s(t)$ takes the value 1 if there is an ongoing surge event at time t, and 0 otherwise (base flow).

Two text files, `flow_name.dat` and `surge_name.dat` should be included in the `flow` folder. Here, `name` is the name of the flow data, which is passed as a parameter to the main executable to identify these files. For the flow file, each line is a value pair t_i v_i. The t_i must be in strictly ascending order. The surge file must have the same number of lines with values 0 or 1 indicating the surge state corresponding to the data in the flow file. The beginning of the simulation corresponds to the initial injection time, so that the onset times should be referred to this time. The first time t_0 must not be greater than the injection time. Any values of t_i smaller than this time occur before the beginning of the simulation and are ignored.

The provided example flow and surge files include a constant base flow $v \equiv 1$ starting at $t = 0$, `flow_example_0.dat` and `surge_example_0.dat`, and a base flow of $v = 1$ starting at $t = 0$ punctuated by a surge event with $v = 10$ from $t = 100$ to $t = 110$, `flow_example_1.dat` and `surge_example_1.dat`.

2 Notes on exponential waiting times

Except in the bed, all transitions between regions are assumed to be characterized by a fixed probability of occurrence per unit time. This is the stochastic equivalent of a fixed rate. We call this probability per unit time a rate also by a slight abuse of language. It follows that the waiting times associated with these events are exponentially distributed, with a mean equal to the inverse of the rate.

The exponential distribution is the only memoryless distribution. As it is characterized by a fixed rate, if a transition hasn't happened by a given time, the waiting time counted from that time has the same distribution as the full waiting time. This allows for a number of theoretical and computational simplifications, which the code relies on. For example, since advection is taken to be piecewise-constant in time, velocity-dependent exponential waiting times may be computed at the onset of a given velocity. If the corresponding waiting time exceeds the time until the onset of the next velocity, no transition occurs for the original velocity and the computation is repeated as needed. Note also that the waiting time to the first of two exponentially distributed events, characterized by rates r_1 and r_2, is the minimum of the two waiting times. It is easy to show that its distribution is again exponential, with rate

$r_1 + r_2$. The mean waiting time is then $1/(r_1 + r_2)$.

Finally, note that the characteristic waiting time to exit from a finite interval of size ℓ by dispersion with a constant dispersion coefficient D is $\ell^2/(2D)$. The waiting time distribution associated with this process is, in general, not exponential, because a particle near the center is likely to take longer to exit than a particle near the edges. However, for times $\gg \ell^2/(2D)$, the exponential distribution with a mean equal to $\ell^2/(2D)$ is a good approximation. For fixed D, the smaller the depth of a region (e.g., the water column) the better this approximation is.

3 Boundary and initial conditions

- The upstream boundary condition is constant-concentration in the water column. This corresponds to a continuous, flux-weighted injection and is implemented by discretizing the injection into constant, user-provided time steps. It starts and ends at the specified times. If the start and end time are equal, an (instantaneous) pulse injection is simulated. At the beginning of the injection, specified amounts of mass may also be injected in the hyporheic region and the bed.

- The downstream boundary condition is absorbing (mass cannot flow upstream).

Note that there is no need to know the residence time upon injection in the water column and hyporheic zone, which are characterized by exponential residence times, which are memoryless. It is important to note that any mass injected in the bed does not account for previous history (recall that bed residence times are not memoryless), and is therefore assumed to be injected there at the time of the initial condition.

4 Processes in the water column

- Piecewise-constant-in-time downstream advection.

- Exponential residence time corresponding to deposition, with an arbitrary dependence on velocity. In the example models provided, during baseflow, this is composed of a baseline contribution to the rate, corresponding to the residence time for $v = 0$, and an additional advective

contribution to the rate proportional to v^2. During surge events, the rate may be different. In the example models provided, the rules are as follows. In models 1 and 3, there is no deposition (zero rate). In model 2, the deposition rate is identical to base flow.

- Decay at constant rate.

5 Processes in the hyporheic zone

- Exponential residence time corresponding to either resuspension to the water column or transition into the bed. During baseflow, there are two possible transition mechanisms. The baseline mechanism, valid when $v = 0$, and an additional advective mechanism characterized by a rate proportional to v. The later is proportional to v in the example models provided. The velocity-dependence of the latter rate can be easily changed within the code framework. The transition proceeds according to the first process to occur, and each process is characterized by a probability of resuspension. During surge events, the rates and probabilities may differ. In the example models provided, these rules are as follows. In model 1, advective resuspension is instantaneous and there is no deposition; the baseline-rate transitions stay unchanged. In models 2 and 3, the rules are identical to base flow, but all advective transitions are resuspension events (no advective deposition).

- Decay at constant rate.

Note that dynamics without this intermediate region can be simulated by setting the average dispersive waiting time in this region to zero and the corresponding resuspension probability to 1/2.

6 Processes in the bed

- Arbitrary residence time corresponding to transition to the hyporheic zone. In the model examples provided, the residence time are skewed-Lévy-stable-distributed. The skewed Lévy-stable distribution is characterized by an exponent β charaterizing the power law tail, and a characteristic time of onset of the power law tail. Note that dispersion

with constant dispersion coefficient in a semi-infinite medium corresponds to $\beta = 1/2$.

- Decay at constant rate.

Note that an impermeable boundary beneath the bed region induces a cutoff time to the residence time in the bed. However, this may be irrelevant if it is very long compared to the times of interest, and also if the decay rate in the bed is sufficiently large, so that *E. coli* are unlikely to survive long enough to reach the boundary and come back. If *E. coli* can drain off into groundwater, there is also both a cutoff time and a probability of never returning, and potentially a source from the groundwater into the bed, which may or may not be relevant.

7 Output

The code produces two types of output: domain, and (optionally) zone. Output data is placed in the `data` folder.

The domain output file name is `Data_EColi_RegionMass_model_modelname_`, where `modelname` is the name of the model being used (see Section 8), followed by the values of each input parameter, separated by underscores, and the file format extension `.dat`. Real-valued parameters are written in scientific notation to three significant figures (e.g., `2.01e+03`). Each line of the output data has the values

$$t \qquad m_w \qquad m_h \qquad m_b \qquad m_a \tag{4}$$

separated by tabs, where t is the measurement time, $m_{w,h,b}$ are the masses of *E. coli* in each region at time t, and m_a is the mass of *E. coli* that has crossed the downstream boundary by time t (i.e., the cumulative breakthrough curve). Note that masses may be converted to *E. coli* numbers by dividing by the mass of one E. coli, or by choosing mass units such that the mass of an *E. coli* is unity. This is controlled by the concentration at the upstream boundary, which can be input as a number or mass concentration. The decay rates are accordingly to be given as number or mass rates. Similarly, concentrations in terms of a unit length are associated with an advection velocity, whereas concentrations in terms of a unit volume are associated with a volumetric flow.

The zone output file name is `Data_EColi_RegionMass_Zones_model_modelname_` followed by the values of each input parameter separated by underscores and the file format extension `.dat` as for the domain output. This output file is only generated if the parameter `zones_name` is set. If so, a file `zones_zones_name.dat` should be placed in the output folder. The lines of this file must have the values of the left and right boundary, in terms of downstream distance, where *E. coli* masses by zone are to be measured as before for the whole domain. There can be any number of zones, which may overlap and can be given in any order. Each line of the output data has the values

$$t \quad m_{w,z_1} \qquad m_{h,1} \qquad m_{b,1} \qquad m_{w,2} \qquad m_{h,2} \qquad m_{b,2} \qquad \ldots, \quad (5)$$

separated by tabs, where t is the measurement time, $m_{w,i}$, $m_{w,i}$, $m_{b,i}$ are the masses of *E. coli* within zone i in each region at time t. Note that masses may be converted to *E. coli* numbers by dividing by the mass of one E. coli, or by choosing mass units such that the mass of an *E. coli* is unity. This is controlled by the concentration at the upstream boundary, which can be input as a number or mass concentration. The decay rates are accordingly to be given as number or mass rates. Similarly, concentrations in terms of a unit length are associated with an advection velocity, whereas concentrations in terms of a unit volume are associated with a volumetric flow.

8 Compiling and executing the code

The code is written in C++. It is header-only except for the main executable, which must be compiled locally using a compiler compatible with C++17. All necessary header files are either provided or part of the C++ Standard Template Library (STL). The following instructions are for a UNIX system (mac or Linux). The compilation procedure consists simply of compiling `EColi.cpp` into `EColi.o` and linking the latter into the final executable `EColi`, and it may be adapted to a windows machine.

8.1 Compilation

Open a command line, `cd` into the `src` directory, and run `make EColi`. This should produce the final executable `EColi_model_modelname` in the `bin`

folder, where `modelname` is the name of the current model defined by the declaration

```
using namespace ecoli::model_modelname;
```

inside `EColi.cpp`. By default, `Makefile` uses the gnu compiler g++. If you prefer to use a different compiler or have g++ under a different command name, edit `Makefile` to set the variable `CC` to the name of your compiler of choice.

A make.sh shell script for automatic compilation given a model choice is also provided in src. The shell script may need to be given executing permission, which can be achieved by running

```
chmod +x make.sh
```

Each model is compiled into a dedicated executable as explained above. Running

```
./make.sh modelname
```

will produce the executable `EColi_modelname`.

8.2 Execution and parameters

The code is run from a command line by executing `EColi_model_modelname` for a given choice of model name, followed by parameter values, separated by spaces. If the executable is run with no parameters, a list of the required parameters followed by short descriptions is printed. Alternatively, parameter values may be placed in a text file, so that records of different parameter sets can be easily kept. For example, to run the code with the provided example parameters file `parameters_example_0.dat` and the model named "1", compile EColi for the latter, and then run

```
EColi_model_1 $(< ../parameters/parameters_example_0.dat)
```

Note that this example uses the flow file `flow_example_0.dat` and the the surge file `surge_example_0.dat`, which must be present in the `flow` directory. The example case simulates a conservative pulse injection in the water column, without the intermediate hyporheic zone. Similarly, `parameters_example_1.dat` uses the flow and surge files `flow_example_1.dat` and `surge_example_1.dat`. It simulates a conservative tracer under base flow punctuated by a surge event.

Note that physical units are arbitrary but should be consistent across all quantities. Note also that the number N of particles injected at each injection discretization time step Δt_{inj} does not represent the actual number of *E. coli*. Rather, it controls the Lagrangian discretization of *E. Coli* mass. Each Lagrangian particle represents a collection of *E. Coli*, and the mass it carries evolves according to decay at a rate depending on the region it is in. The mass in the water column at the time of injection t is given by $\int_t^{t+\Delta t_{\text{inj}}} dt'\, c_{\text{inj}} v(t')$, which we approximate as $c_{\text{inj}} v(t) \Delta t_{\text{inj}}$. As a special case, if the initial and final injection times are the same, the parameter c_{inj} is interpreted as a mass and a pulse injection corresponding to that total mass is simulated instead. The number of particles used in each region is proportional to the amount of mass starting in that region, to a total N of particles. The mass carried by each particle is the mass in the region divided by the number of particles in the region.