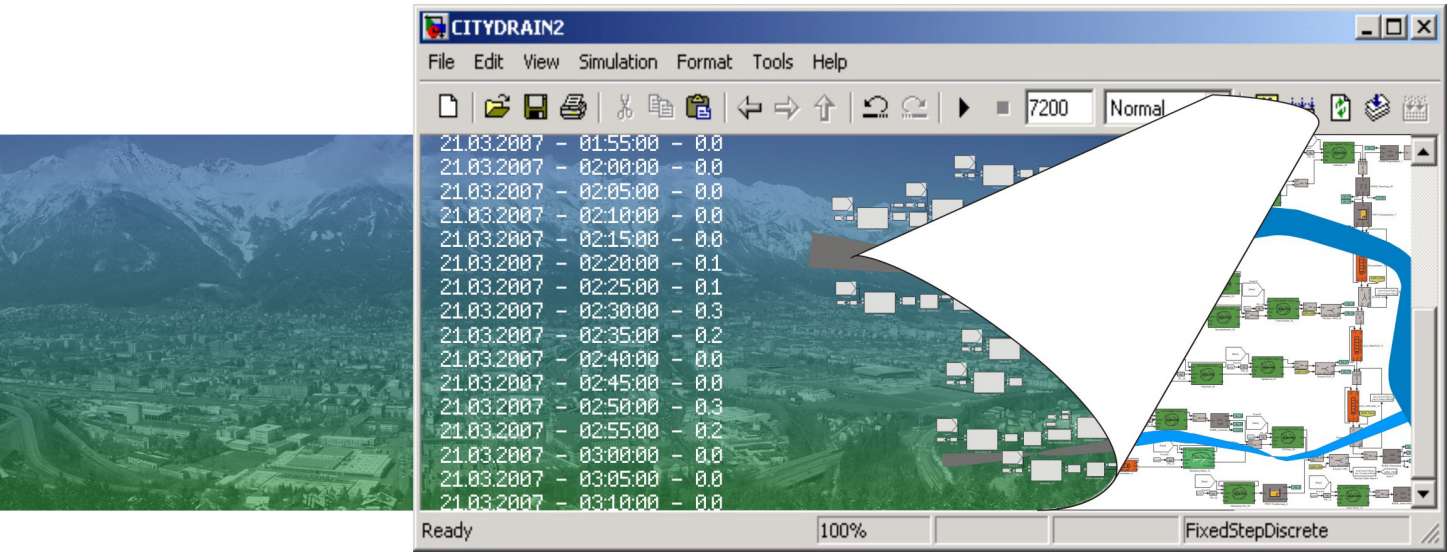
UNIT OF ENVIRONMENTAL ENGINEERING - INSTITUTE OF INFRASTRUCTURE
faculty of civil engineering - university of innsbruck



CITY DRAIN II
INTEGRATED URBAN DRAINAGE

Authors:
S. Achleitner and W. Rauch

Supported by the Fund for Scientific Research established by the state of Tyrol/Austria.
(Wissenschaftsfond des Landes Tirol).

IMPRINT

CITY DRAIN 2.0 – an open source Matlab/Simulink library for integrated simulation of urban drainage systems. (2007)

Software requirements: MATLAB Release 13 (or higher)+Simulink

Availability: *http://umwelttechnik.uibk.ac.at/ or contact developers*

The software is a freeware and may be downloaded at *http://umwelttechnik.uibk.ac.at/*. A copy may as well be obtained by contacting the Institute of Environmental Engineering.

Authors:

Dipl.-Ing. Dr. techn. **Stefan Achleitner**
Univ.-Prof. Dipl.-Ing. Dr. techn. **Wolfgang Rauch**

Development/Programming:

Dipl.-Ing. Dr. techn. **Stefan Achleitner**
Dipl.-Inf. **Heiko Kinzel**
Dipl.-Ing. **Michael Möderl**

IUT – Unit of Environmental Engineering - Institute of Infrastructure
Faculty of Civil Engineering, University of Innsbruck

Technikerstraße 13, 6020 Innsbruck, Austria
http://umwelttechnik.uibk.ac.at
umwelttechnik@uibk.ac.at

Acknowledgements:

The authors gratefully acknowledge the support granted by the Fund for Scientific Research established by the state of Tyrol/Austria. (Wissenschaftsfond des Landes Tirol).

Disclaimer

CITY DRAIN 2.0
an open source Matlab/Simulink library for integrated simulation of urban drainage systems.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Preamble
The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.
When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.
To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.
For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.
We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.
Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.
Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.
The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program

itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)
These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical

distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
END OF TERMS AND CONDITIONS

CONTENT

1 INTRODUCTION

1.1 General Purpose of Integrated Modelling

The aspect of improving ambient water quality, based on the overall management of river basins gained importance during the last years (Blöch, 1999). The emphasis is being put on the improvement of the receiving water quality as well as on the overall management of river basins as requested also by the European water framework directive (WFD). Both aspects require a change in the design procedures for urban drainage systems. Reason is that the application of design rules based on emission criteria does not necessarily lead to an improvement of the water quality – at least they are limited to a certain extend (Lau et al., 2002; Lijklema, 1995). Thus a shift is recently experienced from end of pipe design criteria to ambient water quality approaches (Achleitner et al., 2005). For the application in practice software tools are required that are capable of modelling urban drainage systems (including the receiving water) in an integrated manner. Rainfall as the elementary input source is of irregular occurrence in intensity and duration, which leads to the need of long term simulations for being capable of a systems performance.

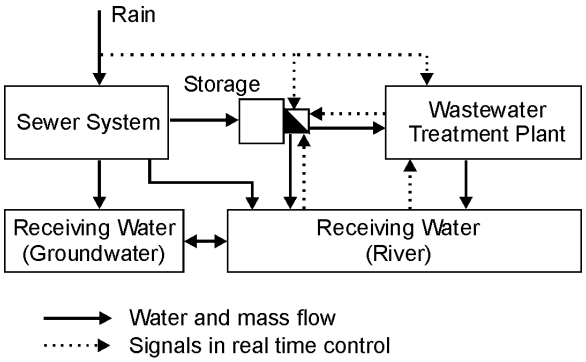


Fig. 1. Schematic on the main elements and information flow in an integrated model (redrawn from (Rauch et al., 2002))

The schematic in Fig. 1 illustrates the main elements and information flow occurring.

1.2 Dominant processes and complexity of models

1.2.1 Principals

A software for integrated modelling may incorporate a variety of models covering hydraulics, mass transport, processes for conversion of matter etc. within the subsystems. Main objective is the prediction of the system performance including the receiving water quality. For choosing the appropriate models it is therefore vital to characterise the impacts onto the receiving water with regard to their type (hydraulic, chemical, bio-chemical,…) and duration (e.g. acute, delayed, accumulating).

Regarding the time scale for modelling not only the dynamics of the relevant processes in the drainage system itself are to be considered but also the duration of the impacts (and associated processes) in the receiving waters. E.g. acute pollution occurs instantly and requires short term modelling whereas accumulative effects in the receiving water can only be covered within a long term simulation effort. But also the stochastic nature of rainfall as the source of impacts in an urban catchment needs to be considered. Single rain events are often source for acute effects in the receiving water such as hydraulic stress or pollutants entering the receiving water. The assessment of those is based on an evaluation of frequency, magnitude and duration of the impact (see. e.g. Harremoës and Rauch (1996)) and thus requires a statistical interpretation. This again is possible only within the framework of long term simulation studies.

Overall the computation in CITY DRAIN © is based on an fixed discrete time steps approach where each subsystem uses the same time increments, usually being predetermined by the timely resolution of the rain data used. Models implement for hydraulics and mass transport are formulated for discrete time steps Δt.

1.2.2 Computational aspects for hydraulics

Flow of water in both sewers and rivers is described by the continuity and momentum equations. The latter is known as the Navier-Stokes or Reynolds equation. The actual form of a hydrodynamic model depends on assumptions made on characterizing turbulence but for water quality purposes mostly the well-known, cross-sectionally integrated (1D) Saint Venant equations or approximations to these equations are used. Different levels of simplifications of the momentum equation are known for describing unsteady flow. Most simple approximation is the kinematic wave model being valid where backwater effects are negligible. All hydrodynamic equations have in common that they are demanding from a computational point of view. There for a variety of simpler conceptual models where developed (frequently denoted as hydrological models). These as well respect conservation of mass but use conceptual relations instead of momentum equations. The rapid simulation with conceptual models puts them in favour to hydrodynamic models regarding computational effort. Effects such as pressurized flow or backwater effects cannot be covered. For allowing long term simulations the blocks implemented in CITY DRAIN are based on purpose on simple conceptual models for hydraulics.

1.2.3 Computational aspects for transport and conversion of matter

For limiting the effort of modelling only relevant pollutants and processes need to be considered. Neglecting issues of secondary importance is required to avoid unnecessary complexity of models. Transport models describe in principle only the flow of soluble and conservative matter through the system. Effects such as physical or biological conversion processes (sedimentation, degradation, etc) are considered by extension of the transport equations.

1.3 Why realising CITY DRAIN in Matlab/Simulink

Basic idea was to create an open source toolbox for integrated modelling of urban drainage systems. For the use in the daily engineering work such software tools are required to be simple in handling and to provide a certain flexibility to be adjustable for different scenarios. Different subsystems should be freely arrangable and connectible to each for describing an integrated urban drainage system and the fluxes of water and matter (Achleitner, 2006).

The principle of block-wise modelling of integrated systems in CITY DRAIN has been developed in a Matlab/Simulink© environment. The platform is widely used for all different kinds of dynamic simulations and was found suitable as hosting environment for the CITY DRAIN© software. On the one hand the platform is tailored for dynamic and time dependent simulations, on the other hand a graphical user interface is already provided.

The user interface is block oriented for convenient usage and creation of coupled models. Blocks are connected to each other providing information flow between each other. Besides using pre-existing blocks provided by Simulink the creation of own blocks is supported. Creation of own routines is done by coding in either m-functions, s-function or C++. For simulation either continuous or sampled (discrete) time may be used. Results can be visualized directly in Simulink. Alternatively results may be stored in the Matlab workspace for visualisation or further analysis.

2 FIRST STEPS

2.1 Installation of City Drain

City Drain requires two simple steps prior being available within Matlab/Simulink environment. Following files are part of the City Drain software:

CityDrain02.zip
CD2_startup.m
CD2_ UserManual.pdf

The file (*CityDrain02.zip*) contains the software library and all associated. Data is to be unzipped and saved preferably in the operating system's programs directory.

C:\Programme\CityDrain02 for German operating system
C:\Program Files\CityDrain02 for English operating system

For convenient use of City Drain 2.0 it is required to include the *CityDrain02* directory (and all subdirectories) in the Matlab paths. Therefore the Matlab "startup.m" file is extended for automatic adding of City Drain directory to the Matlab path.

File:
C:\Programme\MATLAB6p5\work\startup.m

In case there is no startup.m file created in your Matlab, please create a new startup-file. Following code to be added can be found in *CD2_startup.m.* The user may modify the path of City Drain included in the code (bold printed).

```
% Path setting for CITY DRAIN 1.0
% IUT Institute of Environmental Engineering

cd02path='C:\Program Files\CityDrain02';
cd02path_full=genpath(cd02path);
k=strcmp(cd02path_full,'');
disp('Matlab-path for CITY DRAIN 2.0:');

if k==1
     disp('HAS NOT BEEN SET !!');
     disp('Please check in startup.m if path is set correctly.');
     disp(' ');disp(' ');

else
     disp(cd02path);
     path(path,cd02path_full);
     disp(' ');disp(' ');
end

clear('cd02path');clear('cd02path_full');clear('k');
```

2.2 The City Drain Library

To open the City Drain block library type

```
> citydrain
```

in the Matlab command window. Alternative, the Library can be opened via *"File/Open…"*:

```
C:\Programs\CityDrain02\CD2_Library.mdl
```

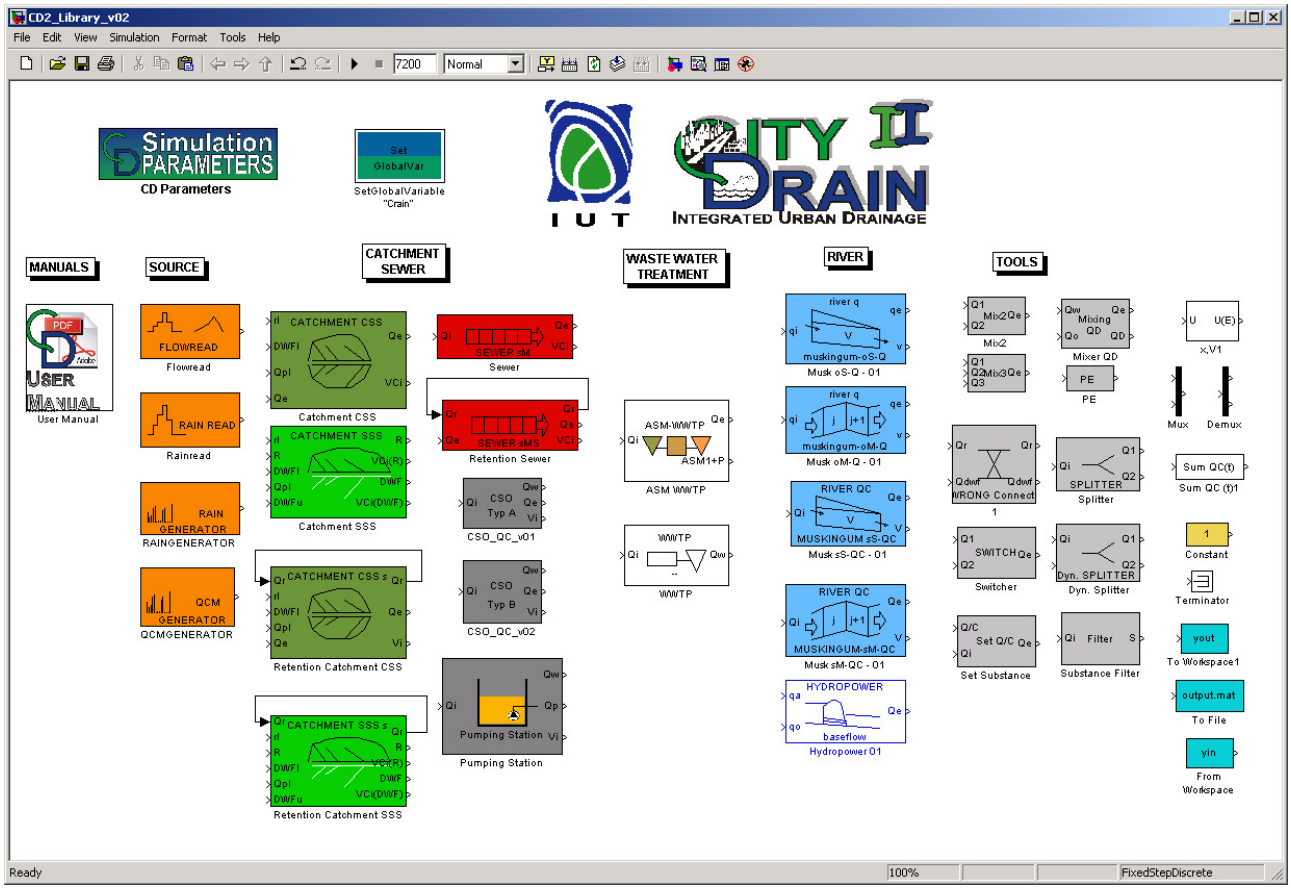


Fig. 2. City Drain 2.0 Block Library (*CD2_Library.mdl*)

The library contains blocks in 5 sections. Core block required for every simulation is the "CD Parameters" blocks organizing global setting for each simulation.

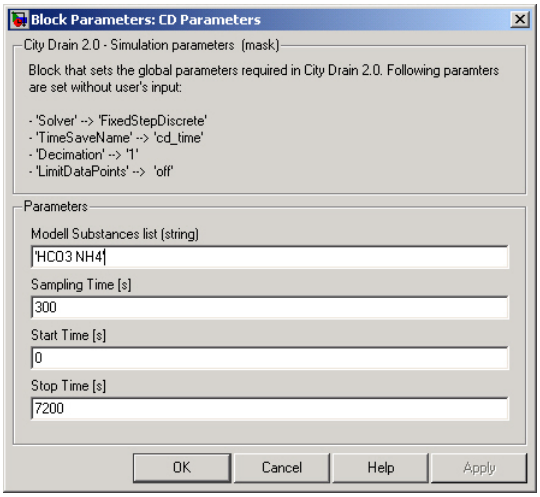
This manual as well as the Tutorial may be opened via double click on the "Manual Blocks". The remaining blocks represent different parts of the urban drainage system and are described in detail in this manual. How to create a new scenario, perform simulations and cope with simulation results is shown in the Tutorial Manual.

3 CONCEPTS AND PROGRAMMING

3.1 Simulation parameters and Unit Conventions

CITY DRAIN and the library blocks implemented are designed to work within a discrete time scheme. Constant and discrete time steps are used within a simulation where simulation time and size of time steps are to be chosen by the used.

Core element of every CITY DRAIN simulation is the block **"CD - Simulation Parameters"**



This block ensures that simulation parameters in the Matlab/Simulink © are defined correctly. User input is required for the

- sampling time Δt,
- start time t_0 and
- stop time t_E.

of the simulation.
In addition to version 1.0 of CITY DRAIN, a global list of substances is to be set in the parameters block. This list is available for all other blocks and can be addressed by using different provided Matlab functions. Utilisation of functions can be done within the mask or as well inside underlying s-function codes of the different blocks. The available functions are described below.
The sampling time defined is utilized within all CITY DRAIN blocks provided, thus is being globally used. Hidden settings (without required user input) are made for

- 'Solver' ..... 'FixedStepDiscrete'
- 'TimeSaveName'..... 'cd_time'
- 'Decimation' ..... '1'
- 'LimitDataPoints' ..... 'off'

THE BLOCK "CD – SIMULATION PARAMETERS" IS TO BE INCLUDED WITHIN EACH SIMULATION TO ENSURE CORRECT SETTING OF VARIABLES.

Convention regarding units in City Drain are as followed:

Q [m³/s]..... Flow [cubic meter per second]
V [m] ..... Volume [cubic meter per second]
L [m] ..... Length [meters]
t, Δt [s] ..... Time [seconds]
C [g/m³] ..... Concentrations [gram per cubic meter]
M [g] ..... Mass [grams]

One mayor improvement of City Drain 2.0 compared to City Drain 1.0 is the substances transportation system. As in City Drain 1.0, data on flow and substance concentration is to be transferred between blocks in each time step. This is done within the Simulink-system on which City Drain is built on. The data is to be transferred as vector from one block to another in a "downstream" manner. The general concept of the vector to have the flow entry at first followed by substance concentrations has been kept.

$$Q = [q, c1, c2 \dots cn]$$

In Version 1.0, the user was required to know which entry of concentration represents what substance, having no possibility to link a substance name to the numerical entry given. The new, respectively improved concept in this version, allows to link names helping to avoid mistaken build up of the model with respect to definition of the concentration vector. Secondly it allows an efficient and more "visible" administration of the substance flows.

Each transfer of data from one block to another is then done using this specific format. The definition of the order of substances is done within the mask of the parameters block using the field "model substances".

Example:

'Ntot COD BOD5 Cu'

is treated as

$$Q = [q, C_1, C_2, C_3, C_4] = [q, C_{Ntot}, C_{COD}, C_{BOD5}, C_{Cu}]$$

The substance names are to be entered as string and are to be separated by blanks from each other. Names given are case sensitive (e.g. 'Ntot' is not equal to 'NTOT'). This list of names is the available to be used in the blocks.

The contrasting alternative to this scheme would be to have a system allowing to have a differing order of substances in each link, requiring a more attention in the maintenance of the blocks. Especially when adding a substance to the system, each block would require to get the information that a substance is added separately. This would, in the extreme case, require to edit all blocks in a modelled scenario.

Using the functions described below, information about the substance list (such as the number of substances included) can be obtained and transferred to each block in an automated manner.

Secondly, assessing a substance concentration for performing calculations can as well be done by a using the substance's name and using one of the below described functions. Advantage is, that the calculation is still done with the correct substance although the number and order of substances is changed. As long as the name given is kept, the correct numerical value of the concentration is used.

Thus, this scheme is to be seen as an highly recommended option but not as a must to be used. Depending on the case, the advanced user may decide to exclude single blocks from the overall scheme simply by not applying the below described functions in that specific block. Secondly, this allows to reused already set up models with version 1.0 of CITYDRAIN.

The global variable "substances" stores the list of all substance in the model as a string. For the below given example to usage of the functions provide the before given example is used.

3.2 Implement functions for substance handling

3.2.1 Function "get-sub"

Function:
c=CD2_get_sub(elec, u)
Inputs:
elec String containing the names of the wanted substances (e.g. 'Ntot BOD5')
u Data vector organized according to global naming scheme
Output:
c Vector of the wanted concentrations in given order according to "elec"
c = [.C_{Ntot} C_{BOD5}]

3.2.2 Function "get_sub_no"

Function:
[s,r] = CD2_get_sub_no(elec)

Returns – for a given string of substance names – a vector s and r containing the concentrations and positions within the global substance vector respectively.
Input:
elec String of substance names (e.g. 'Ntot BOD5').
Output:
s Concentrations of substance in the order as given in "elec"
s= [C_{Ntot} C_{BOD5}]
r Position of substances in the global order substances
r= [1 3]

3.2.3 Function "get_substance"

Function:
concentration=CD2_get_substance(substances_vector,substance)

Grabs from a vector with substance concentrations the correct entries according to the substance names stated. The concentrations are addressed according to the order given in the global substance name
Input:
substances_vector vector with concentrations
substance string of the wanted substance's name
Output:
concentration Vector of wanted substance concnetrations.

3.2.4 Function "get_substance_count"

Function:
n=CD2_get_substance_count(vect)
Returns the number of substances that are given in a string.
Input:
vect string with substance names (usually used with the global variable 'substances'
Output:
n number of substances

3.2.5 Function "get_substance_name"

Function:

name=CD2_get_substance_name(vect, pos)

Returns the name of the substance located at the position "pos" in the string "vect" which contains substance names.

Input:

vect	String of substance names
pos	position of the wanted substance

Output:

name	String of the substance name wanted

3.2.6 Function "get_substances_count"

Function:

n=CD2_get_substances_count()

Input: none

Output:

Number of substance given in the global substance list

3.2.7 Function "permute"

Function:

y=CD2_permute(x,r)

Permutation of values in vector x according to the (new) order given in vector r. The resulting permutation vector is stored in y.

Input:

x	Vector of substance concentrations
r	Vector of new positions for permutation

Output:

y	Vector of substance concentrations in new order

Example:

x=[7 23 40]; r= [2 1 3]; → y= [23 7 40] ;

3.2.8 Function "set_substance"

Function:

subs_out =CD2_set_substance(s_name, s_name_set, concentration)

Generates a zero value vector and sets the concentration of matters given in *s_name_set.*

Input:

s_name	String with substance names given for the target vector (the one to be modified)
s_name_set	String with substance names to be set
conc	Vector of concentrations of substance to be set (order according to string s_name_set)

Output:

subs_out	Generated vector of substance concentrations

3.3 State space modelling and S-functions

For general issues on creating and modification on blocks, the user is asked to stick to the very detailed manuals provided by the Mathworks Company. Still, as this software is open source and it is wanted to have users that extend the library with new and groundbreaking blocks, we would like to give some guidance on how state space modelling works, how numerics are implemented and what most important issued are to known when starting to modify or to create s-functions.

Realisation of a block wise representation of parts of the urban drainage system as in Rauch *et al.* (2002) is based on the state-space approach as shown by (Schreider *et al.*, 2001). The principles of state-space modelling have been formulated by (Kalman, 1960) and it is since then widely used. The Matlab/Simulink© environment is using the same principles embedded in a graphical user interface to arrange blocks (The Mathworks, 2003). In principle input (*u*) is fed to a state-space model that is variable in time. A dynamic output is generated, based on both, the dynamic input (*u*) and the model's state (*x*).

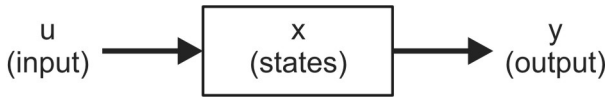


Fig. 3: State-space model

The state (*x*) of the system is defined as the values of state variables at any instant point of time. When modelling urban drainage systems this may be the current volume *V* or the current substance concentration *C*. The utilization of states is the core element, where the change of the states is defined via mathematical equations. This can be either differential equations or – as used in CITY DRAIN – discrete formulations of the differential equations. As an additional information, the state-space model accepts parameters which are constant in time. A linear state-space model is described by the following equations

$$\frac{dx}{dt} = Ax + Bu$$
$$y = Cx + Du$$

The set of differential equations describing the output as function of states and input are to be solved for each time step. Different numerical methods for solving the differential equations are available. The time steps for which the equations are solved relate to the accuracy of the solution. Depending on the software, the time steps can be implemented as variable or – as in CITY DRAIN – as fixed time steps. Usually when implementing a dynamic system, the time management is to be organised by the programmer. Within the Simulink environment, a defined scheme is already provided that takes care of the time management. Thus, discretised equations are to be formulated for a single time step and to be embedded in the provided scheme.

The scheme is realized in so called s-functions that are available for different software languages (Matlab, C++ or MEX functions). Within the graphical environment each block addresses a certain s-function. The following figure shows the temporal workflow of initialization, output and update procedures (Achleitner, 2006).

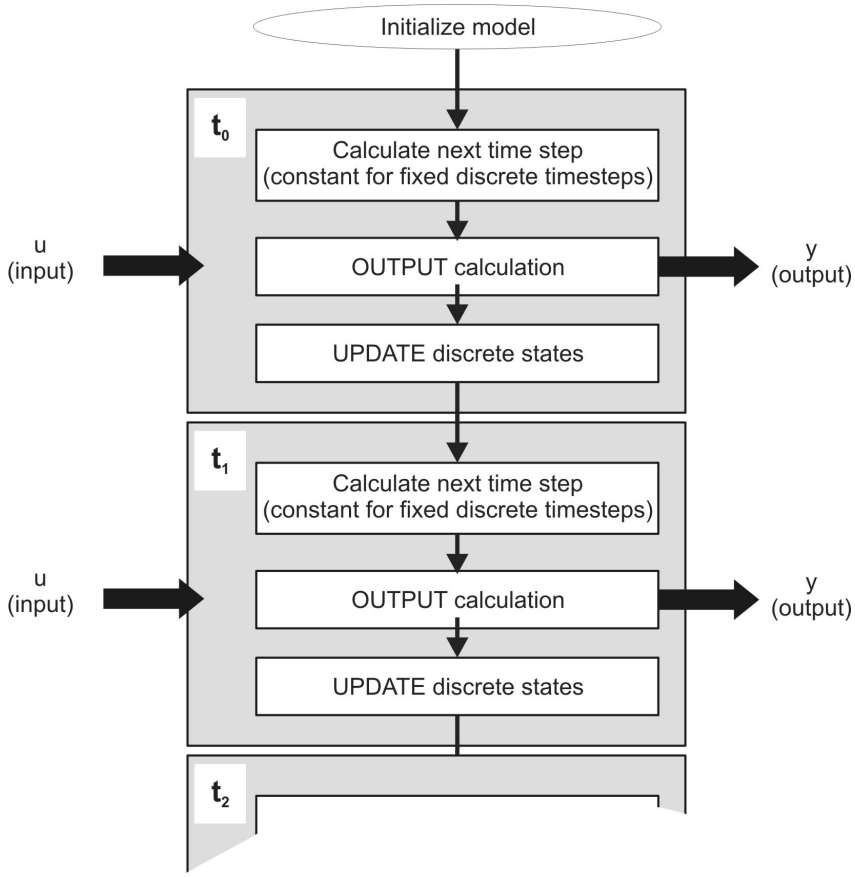


Fig. 4: Simulink workflow for Initialization, Update and Output

When the model is started, the defined state variables are to be set within the initialization stage. This initial setting of the state variable is used within the first output calculations at *t = 0*. Thus, one can virtually attribute the initial states to the time step *t = -1*. Subsequently, output and update procedures are redone for each time step. This calling of different routines is described within the Matlab/Simulink help as a set of *S-function callback methods*. The Simulink environment is thereby calling the appropriate methods automatically while running a simulation. The S-function methods included are

Initialization
of the S-function to create *SimStruct*, which contains the information about the S-function. The main tasks are to set the number and dimensions of inputs, a outputs and states. Further the sampling time is to be sets.

Next sample hit calculation
To calculate the next sampling time step which is only required for variable sample time blocks. This option is not used withing CITY DRAIN.

Output calculation
generating the outputs at the major time step.

Update of states
to store the current state at the major time step for having it available at the next time step as calculation basis.

In the following only the most important issues of s-function using discrete time steps and Matlab m-file language are discussed. For getting to know the full spectra of possibilities in coding Simulink s-functions the user may refer to the respective manuals on s-function provided by Mathworks.

To get to know s-functions “linewise”, guidance is made using the the s-function

```
CD1_sfun_Muskingum_sM_QC.m
```

as an example. The function contains all relevant sections that are required for an s-function.

In general s-functions are nothing mysterious compared to usual m-functions. In fact they are “just” m-functions that are to be set up in specific way so they can be used by the Simulink environment in a standardized manner. Main difference to regular programming is, that Simulink takes over the time management and coordinates thereby the dynamic behaviour of blocks operated simultaneous.

As known from Matlab m-function, the function declaration is to be the first entry.

```
[sys,x0,str,ts] =
CD1_sfun_Muskingum_sM_QC(t,x,u,flag,tstep,K,X,CA,CB,n_comp,N)
```

The left hand side arguments returned are standardized and shall not be changed. In the same way, the first arguments (t,x,u,flag) passed to the function are not be modified as they are the minimum requirement to properly run the s-function. The remaining ones are additional parameters that are specific for this function.

The parameters passed on represent partly dynamic and partly static parameters.
t,x,u,flag are generated and passed automatically by Simulink environment, thus they are variable over time.

t	…current time
x	…the current state of the block
u	…the input to the block
flag	…is mode at which the s-function is currently called

Within the states (x) the actual condition(s) such as the currently stored volume or currently given concentrations etc. are stored.

Parameter u contains the dynamic input fed to the block/s-function at the current time step. Unfortunately the s-functions accepts only one dynamic input being a row vector. Therefore, if blocks are required to be subjected to two or more dynamic inputs, these streams are to be merged to one row vector (see Mux and Dmux blocks in Simulink) prior fed to the s-function block. Once u is passed over, the row vector may be taken apart for convenience in performing calculations. As additional information the number of components (n_comp) and the number of inflows expected(N) are passed to the function. The both allow to extract the correct part of the vector for further calculations. In the following example 2 streams are expected with n_comp component each.

$u=[q_A\ C_{A,1}\ C_{A,2}\ \dots\ C_{A,n\_comp}\ q_B\ C_{B,1}\ C_{B,2}\ \dots\ C_{B,n\_comp}]$

$u(1\ ,\ n\_comp+1)= [q_A\ C_{A,1}\ C_{A,2}\ \dots\ C_{A,n\_comp}]$

$u(n\_comp+2\ ,\ 2*(n\_comp+1))= [q_B\ C_{B,1}\ C_{B,2}\ \dots\ C_{B,n\_comp}]$

Per definition, CITY DRAIN deals with q being the mean flow entering or leaving a block during the last time step (similar conventions are used in recording rain series). This means, when the s-function is called at t=1800sec and the timestep is tstep=300s, the flow is the mean inflow between 1500sec and 1800sec. Analogous this convention is applied for dealing with concentrations.

In contrast states (x) are attributed to a discrete point in time. Thus, a stored volume or concentration is the state at t=1800s at the end of the past time step.

Having this convention in mind, derivation of discrete formulations from differential equations are made accordingly. This scheme led for the applications implemented so far to more simple and numerically stable numerics.

The last parameter required by Simulink is "flag". The s-function is called at each time step automatically. Thereby the “flag” defines the mode at which the s-function is currently run. The implemented switch leads to either one of the functions

- mdlInitializeSizes
- mdlOutput
- mdlUpdate

where other functions than these may be implemented when dealing with continuous models.

```
============================
switch flag,
  case 0,
    [sys,x0,str,ts] = mdlInitializeSizes(n_comp,N,tstep);

  case 2,
    sys = mdlUpdate(t,x,u);

  case 3,
    sys = mdlOutputs(t,x,u,tstep,K,X,CA,CB,n_comp,N);

  case 9,
    sys = []; % do nothing

  otherwise
    error(['unhandled flag = ',num2str(flag)]);
end
============================
```

In this example, flags other than 0, 2 or 3 are not defined and dealt as unhandled flags.

"flag=0" - mdlInitializeSizes
When it is called first time, no states for the function are set yet. When the model is started, the defined state variables are to be set within the initialization step. This initial setting of the state variable is then used within the first output calculations at t = 0. Thus, one can virtually attribute the initial states to the time step t = -1 as outlined before.

Additionally general conditions such as vector sizes used for input (u), states (x)etc. are to be set as well as the timestep size. As the same s-function scheme is used for continuous modelling, a lot more variables can be defined compared to the minimum needed for discrete modelling. The variables of interest are:

```
% Volumes and concentrations only
sizes.NumDiscStates  = N*(n_comp+1);

% Volumes and Flows including corresponding concentrations
sizes.NumOutputs     = 2*(n_comp+1);

% Inflow and concentrations
sizes.NumInputs      = n_comp+1;
```

Passed on information on the number of components (n_comp) and the number of compartments used (N) allow to keep the s-function flexible with regard to the number of substances and subreaches used. The information on n_comp and N are input made by the user within the mask and is passé don via the s-function block. The actual numerical values for the initial states are set in

```
x0  = zeros(sizes.NumDiscStates,1);
```

which is in this example set to zero. If one requires any other starting conditions, modifications may be made at this point. In general, starting with other initial conditions than zero without a good reason is not

recommended. Information on the start conditions are then hidden in the code and may lead to confusion e.g. for mass balancing.

```
==================
str = [];

sys(7)=1;
ts  = [tstep 0]; % driven by global timesteps defined
==================
```

The above are not to be changed as they make CITY DRAIN running with fixed discrete tie steps of size tstep.

```
====================
% Initial setting of values for previous time steps

u_dat.volume=zeros(N*(n_comp+1));

set_param(gcb,'UserData',u_dat);
====================
```

These lines define a global variable u_dat on the blocks level. The gcb returns the handle for the current block (GetCurrentBlock). The u_dat variable is used for exchanging calculation results between sections mdlOutput and mdlUpdate.

Subsequently after initialization is finished, the s-function is called twice repeatedly at each time step, first calling the mdlOutput function (flag=3) and secondly calling the mdlUpdate function (flag=2).

As outlined, any other parameters other than t,x,u and flag can be passed to the s-functions and consequently to mdlOutput or mdlUpdate as they may be required to perform the calculations. The mdlOutput section is designed to generate the dynamic outputs that finally leave the block, where the update section is designed to set the new states (e.g. volumes, concentrations, etc.) obtained in this step. A shortcoming is that the exchange of information between the output section and the update section is limited to be done via a global variable such here used u_dat.

When considering mathematics as given here the calculations of outputs and new states are merged such that one cannot calculate an output without calculating the states as well. Still, stated are not automatically available in the update section as the mdlUpdate function is run when the s-function is second time called in this time step. To have the states available in the update section as well (in order to store them properly) two options are available:

- Doing the calculations a second time or
- transferring the values from mdlOutput to mdlUpdate via a temporal storage in a global variable such a u_dat.

Neither one of the two option is strictly recommended. It depends more on the number of calculations performed. In this example the first option was used, perfoming first all the calculations in the output section and storing the states calculated in the global variable u_dat at the end of the function.

```
======================
% renewing the storage vector u_dat for next step usage

u_dat.volume=xnew;
set_param(gcb,'UserData',u_dat);
======================
```

Finally calculated outputs are passed to the Simulink environment at the end of the mdlOutput function via the return variable sys:

```
============================
% Generating output for block
out=[QE, V];
sys = out;
============================
```

In contrast to the mdlOutput function the mdlUpdate function is kept puristic.

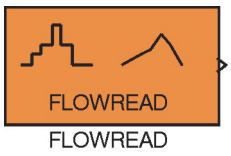
```
======================
function sys = mdlUpdate(t,x,u)

u_dat=get_param(gcb,'UserData');

VC=u_dat.volume;

sys = [VC];
======================
```

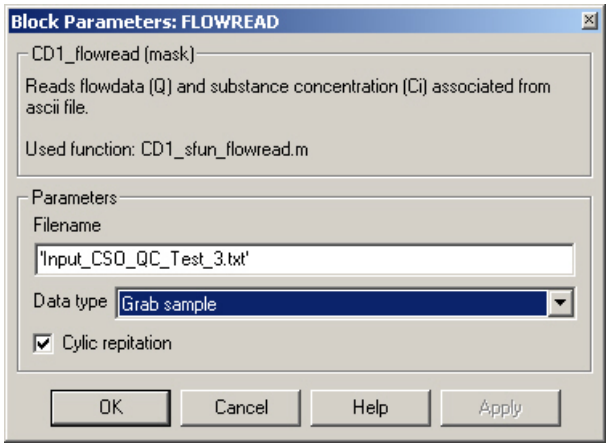
No additional parameters other than (t,x,u) are passed and the function is limited to read the global variable and pass it to the Simulink environment via the return variable sys.

In some s-functions one will find the mdlUpdate function missing (e.g. mixing blocks). This is simply due to flow enters and is instantly passed on. As no storage is required, no states and consequently no mdlUpdate section is needed.

4 SOURCE BLOCKS

4.1 Flowread





Function:

Reads flow data from ASCII files containing time t, flow Q and concentrations C as input. Data is to be provided column wise. First row in the file allows to hold an alpha-numeric descriptor for column data.
The time is to be provided in [sec] starting with t=0. Sampling time $\Delta t_{CITYDRAIN}$ in the simulation must not necessarily be same as the sampling time given in the raw data. An automatic interpolation of data is provided.
In case the data set ends before the end of simulation, values are set to zero.
Cyclic repetition of data is optionally provided which may be used for e.g. the repetition of daily flow dynamics within long term simulations. Requirement for cyclic repetition is that the first data set Q(t=0), C(t=0) and the last data set Q($t=t_{max}$), C($t=t_{max}$) are equal.

Data provided may either represent grab samples (measurement at specific point of time) or composite samples (values representing the mean concentration / flow over time).

Input:

none

Output:

Q Flow [m³/s].
Ci Pollutant concentrations [g/m³]
The number of pollutant concentrations is automatically inherited from the ASCII file storing the raw data.

Parameters:

Source file containing the hydrograph Q(t) and pollutograph C(t) (optional).

Data type
"Grab sample"….Flows Q and concentrations C given are measured values corresponding to the specific point of time.
"Composite sample"….Flows Q and concentrations C given represent mean values corresponding to the past sampling period.

Initial Conditions:

No initial condition required

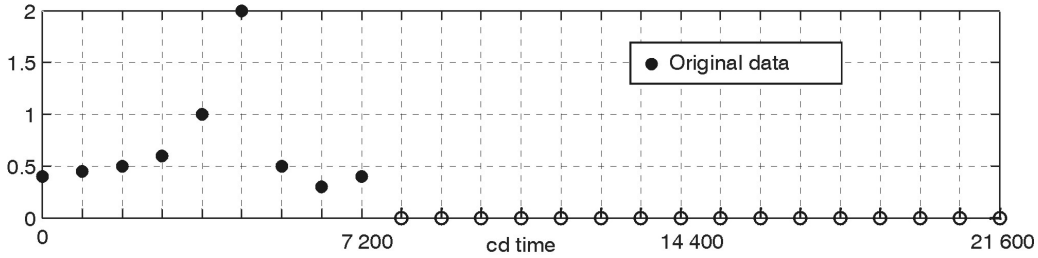
Format / Example Usage:

n the following the output generated by flow read is shown having an example input file containing 15 min values (ΔT= 900 s) of flow and pollutant concentration. Tabulators are used as delimiters.

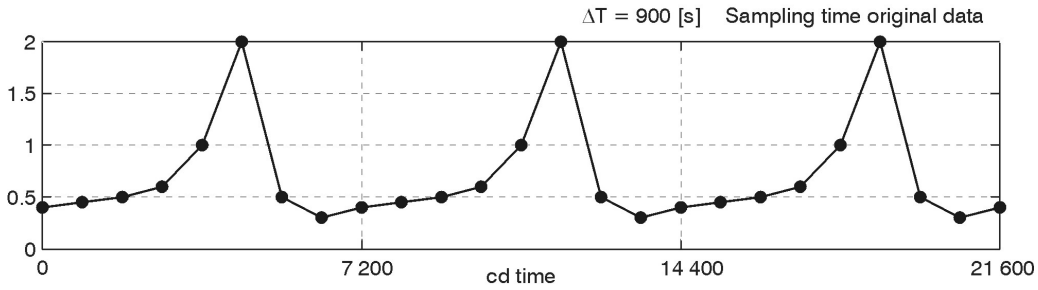
Example input file (*Input_Flowread_Example.txt*):

```
t       q       C1
0       0.40    0.16
900     0.45    0.20
1800    0.50    0.25
2700    0.60    0.36
3600    1.00    1.00
4500    2.00    4.00
5400    0.50    0.25
6300    0.30    0.09
7200    0.40    0.16
```

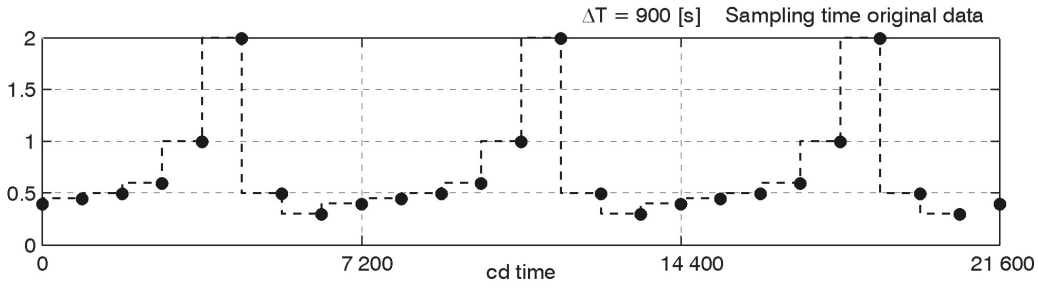
The example input is prepared for usage with the cyclic repetition option. There for the first and last entry have to be equal.



(a) Original Data



(b) Cyclic repetition - Interpretation as grab samples



(c) Cyclic repetition - Interpretation as composite samples

Fig. 5. Data read with cyclic repetition having equal sampling time in data (ΔT) and simulation Δt.

Fig. 5 (a) shows a plot of the raw input data having a sampling time of ΔT=900 [s]. Due to cyclic repetition being applied, the last entry of data is substituted as default by the first data

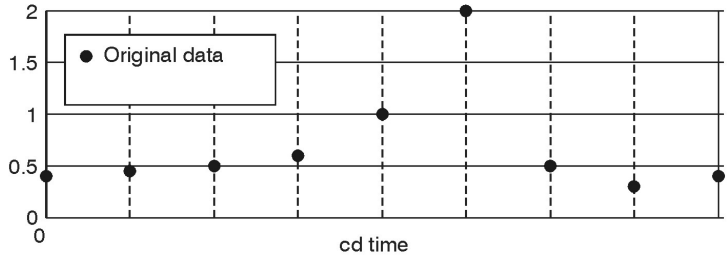
entry. The user is requested to provide data used for cyclic repetition having equal data at the first and last entry.

Interpretation of raw data
The raw data read may be interpreted as

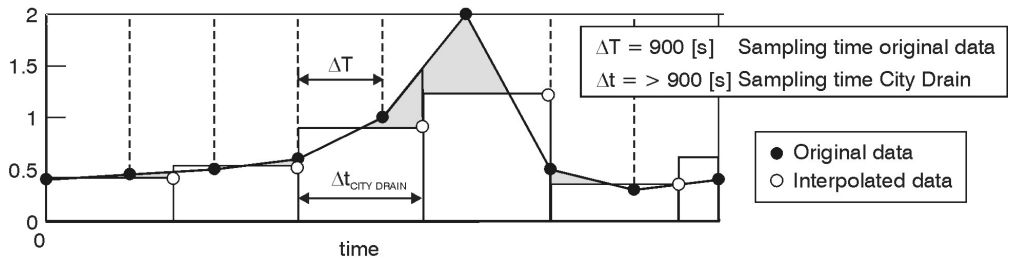
- grab samples (Fig. 5(b)) or as
- composite samples (Fig. 5(c))

For grab sample data, the distribution of flow and concentrations is assumed to be linear over ΔT between data points. In the case of composite samples the data read represents mean flows / concentration over the past time step ΔT.

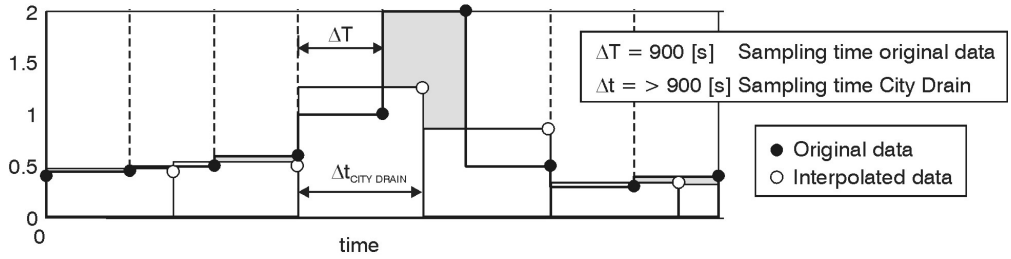
Transfer from raw data to City Drain output data
The output generated from this block is always of type “composite sample” regardless what type of raw data was used. The values represent the mean flow / mean concentration over the last time step $\Delta t_{CITYDRAIN}$.



(a) Original Data



(b) Interpretation as grab samples



(c) Interpretation as composite samples

Fig. 6. Interpolation of data when sampling time given in the raw data ΔT and simulation Δt are not equal.

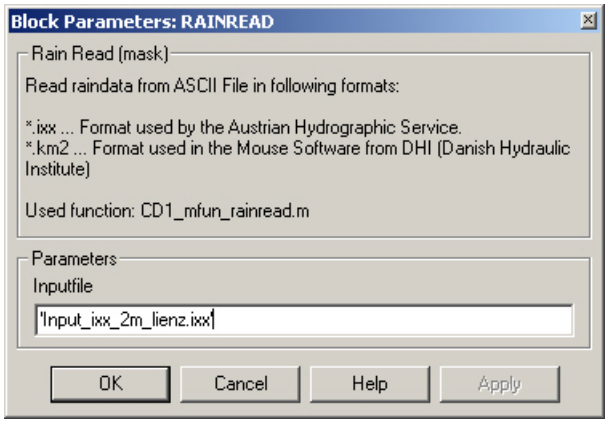
In Fig. 6 (a) a plot of the raw data are shown having time steps ΔT. The example output generated is for sampling times $\Delta t_{CITYDRAIN} > \Delta T$. An internal algorithm is used to account for transferring raw data to output data. Differences in interpolated output is given for raw data being interpreted as either grab sample Fig. 6 (b) or as composite sample Fig. 6 (c) (grey shaded areas).

The algorithm is based on the principle of conservation of mass. Volume (V) and mass flux (F) over each time step are integrated and are maintained when transferred to sampling steps used in the simulation ($\Delta t_{CITYDRAIN}$).

$$V = const. = \int q_{RAW}(t)dt = \overline{q} \cdot \Delta t_{CITYDRAIN}$$

$$F = const. = \int q_{RAW}(t) \cdot C_{RAW}(t) \cdot dt = \overline{q} \cdot \overline{C} \cdot \Delta t_{CITYDRAIN}$$

4.2 Rainread



Function:

Reads rain data from ASCII files having a predefined format. As output rain data is provided following the sampling time of the simulation environment. The dates read are transferred into numerical values of time, where counting of time is started with t=0 at the earliest date/time obtained.
See the formats section for the data types supported.

Input:

none

Output:

r_R Volume of rain per time step in [mm/ΔT].
tout *Run time is virtually transferred to the Simulink environment*

Parameters:

Inp.File Name of ASCII file (e.g. 'filename.ixx') storing the rain data to be read.

Format *For the type of format no additional user input is required. The appropriate format is defined by the file extension. Following extensions are supported:*

- 'ixx'
- 'km2'
- 'mse'

See section *Formats* for details on the type of rain data formats supported.

Initial Conditions:

No initial condition required

Formats:

Supported formats in *CD1_sfun_rainread.m*

IXX Format used by the Austrian Hydrographic Service. The format uses a line of file for one data entry every 5-minute interval. V_R of time interval t_1 to t_2 is numerically attributed to time t_1 . The format of a line is written as:

DD.MM.YYYY_hh:mm:ss_r_R

with
DD (day), MM (month), YYYY (year)
hh (hour), mm (minute), ss (second)
r_R (Volume of rain [mm])

Example input file:

```
01.01.1991 00:00:00        0.1
01.01.1991 00:05:00        0.1
01.01.1991 00:10:00        0.1
01.01.1991 00:15:00        0.1
01.01.1991 00:20:00        0.1
```

KM2 This format is used as well within the software MOUSE from DHI (Danish Hydraulic Institute). In contrast to the ixx format, where dry rain periods are stored as well, this format produces rather small file sizes.
Rain data is stored as well for discrete time steps but splitted into different rain events. Rain events are defined by the dry (rainless) period in between. By default the dry period separating two rain events is taken as 1 hour which is roughly the time for a drainage system to empty. Thus, two consecutive events do not interact hydraulically in the system.

Each rain event is always introduced by a header and then followed by the rain data itself.

Syntax of the header:

3 YYYYMMDD hhmm 0 N t V.V

3	Internal Code
YYYYMMDD	Date
hhmm	Starting time in hours and minutes
0	Internal code
N	Number of intervals recorded in the event [-]
t	Timestep of one interval [min]
V.V	Total rain volume cumulated in the rain event [mm]

Rain data:
The rain data is stored row wise carrying 10 entries in a row. One line is started by two space. Rain data R is written with one integer digit and three decimal digits stored as rain rate having the unit [10^{-3}mm/s]. Conversion from rain rate R to rain volume r per time step is done by

$$r[mm/\Delta t] = R[10^{-3} mm/s] \cdot 0.001 \cdot \Delta t[s] .$$

Example input file:

```
3 19900425 0820 0 1  5   0.1
  0.333
3 19900426 0955 0 18  5   0.4
  0.250 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.250 0.000
  0.250 0.000 0.000 0.150 0.100 0.000 0.248 0.002
3 19900426 1320 0 8  5   0.2
  0.150 0.100 0.000 0.000 0.000 0.000 0.000 0.250
3 19900426 1915 0 2  5   0.1
  0.228 0.022
3 19900427 0925 0 11  5   0.6
  0.293 0.002 0.000 0.000 0.293 0.272 0.319 0.298 0.293 0.002
  0.295
```

Since the output generated from the block is a continuous stream of rain data, timely gaps in between the rain events are filled with zeros for the rain volume.

MSE

MSE formatted rain data only stores rain events, neglecting dry periods. When reading the data timely gaps are filled by zero values for dry the periods.

The format uses a line per data entry having either 5 or 10 minute intervals. r_R of time interval t_1 to t_2 is numerically attributed to time t_1 . The format of a line is written as:

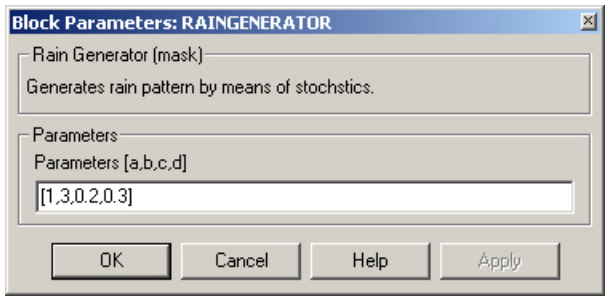
YY_MM_DD_hh_mm_ss_r_R

with
DD (day), MM (month), YY (year)
hh (hour), mm (minute), ss (second)
r_R (Volume of rain [mm])

Example input file:

```
81  1  1 22 40  0  0.0000
81  1  1 22 50  0  0.0000
81  1  1 23  0  0  0.1670
81  1  2  4 30  0  0.1670
81  1  2  4 40  0  0.0000
81  1  2  4 50  0  0.0000
81  1  2  5  0  0  0.0000
```

4.3 Raingenerator



Function:

Generates rain data by means of a simple stochastic algorithm.

Input:

none

Output:

r_R Volume of rain per time step in [mm].
tout *Run time is virtually transferred to the Simulink environment*

Parameters:

a, b, c, d Parameters used for calibration of stochastic processes

Initial Conditions:

No initial conditions required

Theory:

The rain series produced is based on a simple stochastic algorithm for generating the main parameters describing a rain series:

T_{DRY} Duration of next dry period
T_{RAIN} Duration of next rain event
r_M Mean rain volume for the next rain event [mm/Δt]

Within a discrete formulation, the durations T_{DRY} and T_{RAIN} are for sake of simplicity given as the number of time steps. At the end of the last rain event (t_0) the durations and the mean rain volume is calculated as:

$$T_{DRY} = -\frac{1}{a} \cdot \log(\theta_a) \cdot \frac{86400}{\Delta t}$$

$$T_{RAIN} = -\frac{1}{b} \cdot \log(\theta_b) \cdot \frac{86400}{\Delta t}$$

$$r_M = -\frac{1}{c} \cdot \log(\theta_c) \cdot \frac{1}{T_{RAIN}}$$

The parameters a, b and c are used for linear scaling of random numbers generated, thus may be used for calibrating the scheme to local real rain series. The log scaled parameters θ_a, θ_b and θ_c are random numbers being uniformly distributed in the interval (0,1).

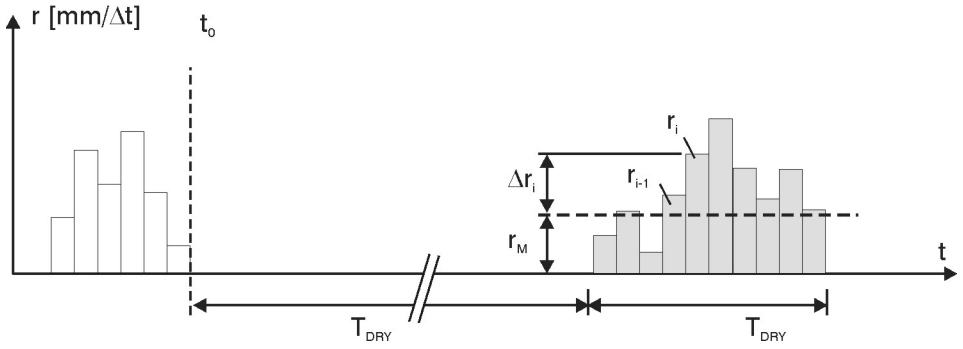


Fig. 7. Schematic for stochastic rain generator

Magnitudes of consecutive rain intensities r_i within a rain event are as well based on a stochastic process. A single rain event is evaluated as deviation Δr_i from the mean rain intensity r_M.

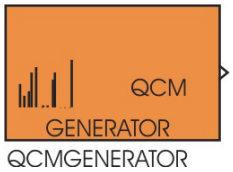
$$r_i = r_M \pm \Delta r_i$$

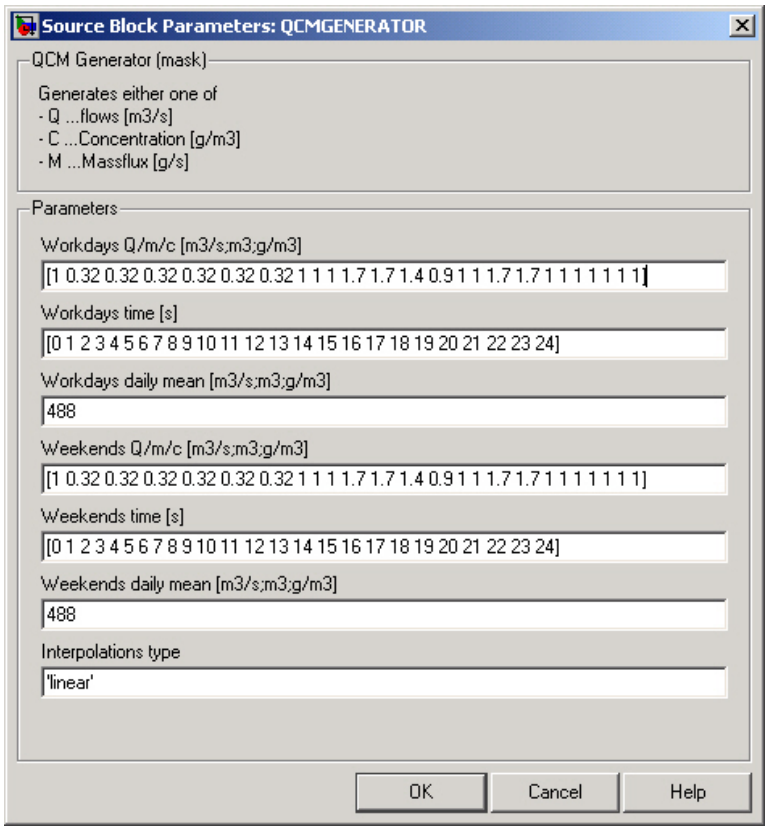
The deviation Δr_i is generated randomly, but using the last deviation Δr_{i-1} generated as an additive term. This is done for including tendencies of increase or decrease and to avoid unnatural jumps in consecutive rain intensities generated.

$$\Delta r_i = \frac{\Delta r_{i-1}}{2} + \Pi \cdot d \cdot r_M$$

The parameter Π used is a random number being normally distributed with mean 0, variance δ^2=1, and standard deviation δ=1. The constant d is used for linear scaling and required as user input.

4.4 QCM-Generator





Function:

Generates a dynamic output (either for flow Q, concentration C or mass M) on a weekly basis. The block is primarily designed to generated dynamic fluxes representing flow of concentration of DWF.

Input:

none

Output:

Q/C/M Dynamic vector alternating after one week; Depending on the user the dynamics represent either flow Q, concentration C or mass M.

tout *Run time is virtually transferred to the Simulink environment*

Parameters:

The parameters Q/C/M, time and daily mean value are to be inserted for workdays and weekends respectively.

Q/C/M Dynamic vector scaled to unity (mean value=1); Depending on the user the dynamics represent either flow Q, concentration C or mass M.

time time vector associated to Q/C/M vector

daily mean Represents the mean daily flow, concentration or mass discharged for workdays or weekends.

Interpolation type: ‘linear’, ‘spline’

Theory:

The parameters Q/C/M, time and daily mean value are to be inserted for workdays and weekends respectively. From these given time series (typically hourly values), time a time

series for a full week is generated (5 workdays and 2 weekend days) and are repeatedly sent as output.

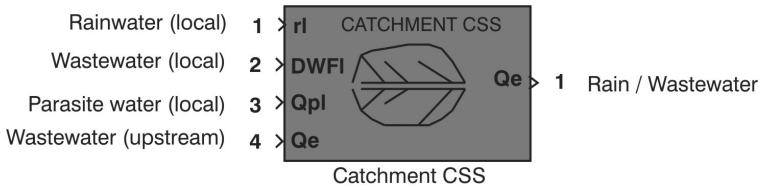
Temporal scales are adjusted to the time steps used in the current simulation where values are interpolated respectively. The type of interpolation made is either ‘linear’ or ‘spline’.

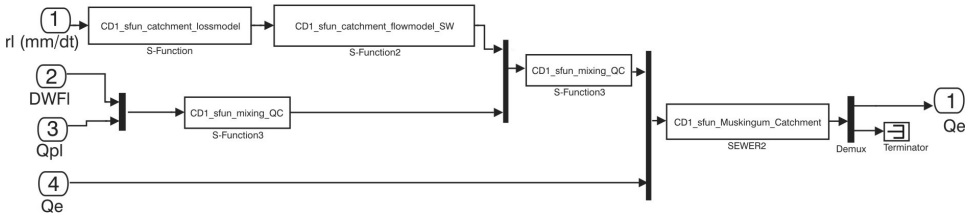
5 CATCHMENT/SEWER BLOCKS

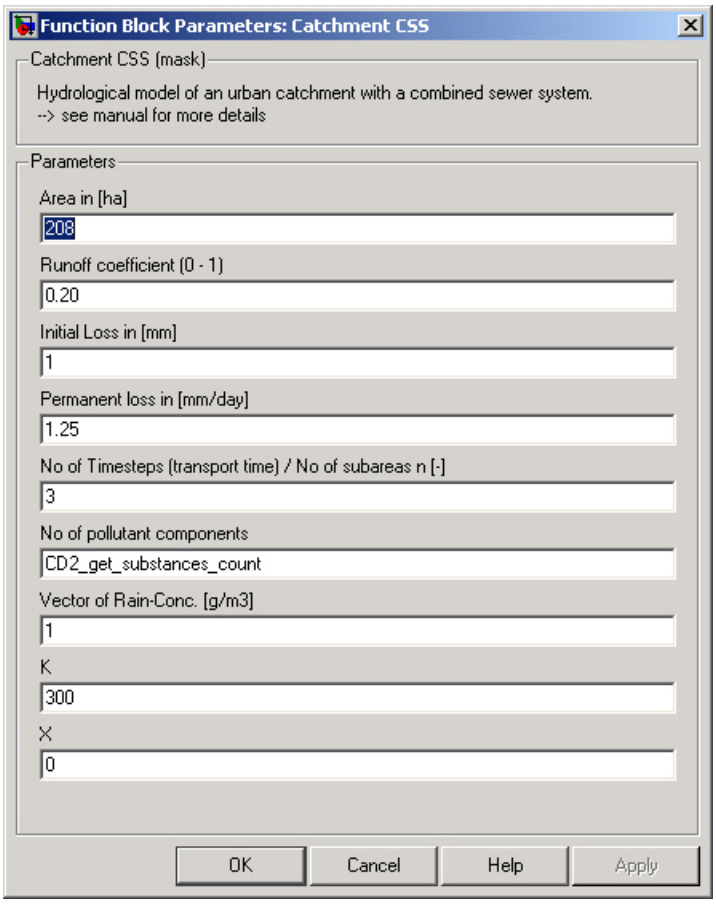
In comparison to CITY DRAIN 1.0, the catchment blocks have been significantly enhanced with regard to dynamics of flow and pollutant transport, Where the former blocks transport mechanisms was based on pure translation, these blocks mimic transport processes by the Muskingum method.

Again, blocks are differentiated for Combined sewer system (CSS) and Separate sewer system (SSS). For both types, a “classic” version using Muskingum exclusively and a “Retention” version that introduced additional storage volume, are provided. The retention-type catchments may be especially used in flat urban catchment where the Muskingum routing method (and its retention/storage ability) may not be sufficient. The Muskingum with multiple subcatchments is used in the catchment blocks as well as in Sewer and River block presented in chapter 5.4.1 Simplified Muskingum .

5.1 Catchment CSS







Function:

Catchment CSS block is designed to simulate a combined sewer system on catchment level. The block copes the major drainage-related processes in an urban area and returns for each time step both the current amount and pollutant concentration of the aggregated outflow from the catchment. Storm water runoff and water quality aspects are computed here with a set of consistently simple conceptual models. The main processes that appear on an urban area in this context are

- Runoff generation
- Overland flow

- Dry weather flow generation
- Runoff and wasteflow pollution

The associated dynamic inputs can be distinguished for inputs that (a) originate from the catchment and (b) inputs which originate from upstream and are to be routed through the catchment.

The modified scheme allows both, feeding of the uppermost block ($Q_{I,U}$) as well a distributed feeding of blocks ($Q_{I,L}$). Thus, inputs provided such as the rain intensities (*rl*) acting on the catchment, the dry weather flows generated in the catchment (DWF_L) and parasite water infiltrating into the sewer system (*Qpl*) are distributed homogeneously within the catchment. Flows from upstream of the catchment are all the way routed through and thus are fed to the uppermost sub-block. In case of the CSS block an upstream wastewater stream *Qe* may be provided as dynamic input.

Input:

r_L	Rain volume per time step [mm/ Δt].
$Q_{DWF,L}$	Dynamic dry weather flow [q [m³/s] C_1[g/m³] C_2[g/m³]
$Q_{P,L}$	Dynamic flow parasite water [q [m³/s] C_1[g/m³] C_2[g/m³]
Q_e	Wastewater/stormwater flow introduced from an upstream catchment [q [m³/s] C_1[g/m³] C_2[g/m³]]

Output:

Q_e	Combined sewer (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³] ….].

Parameters:

A	Catchment Area A [ha]
φ	Runoff coefficient φ [-]. In the range of φ = 0...1.
hi	Initial loss [mm]
hp	Permanent loss [mm/ day]
n_{TA}	Number of sub areas/subreaches n_{TA} [-]
n_P	Number of pollutants n_P [-]
C_{RAIN}	Vector of pollutant concentrations of storm water flow [g/m³]
K	Muskingum parameter describing the time required for a discharge wave travelling through the reach [s]. **K applies to one subreach** and does not cover travelling time for all sub reaches.
X	Dimensionless weighting factor that relates to the amount of wedge storage [-] in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).

Initial Conditions:

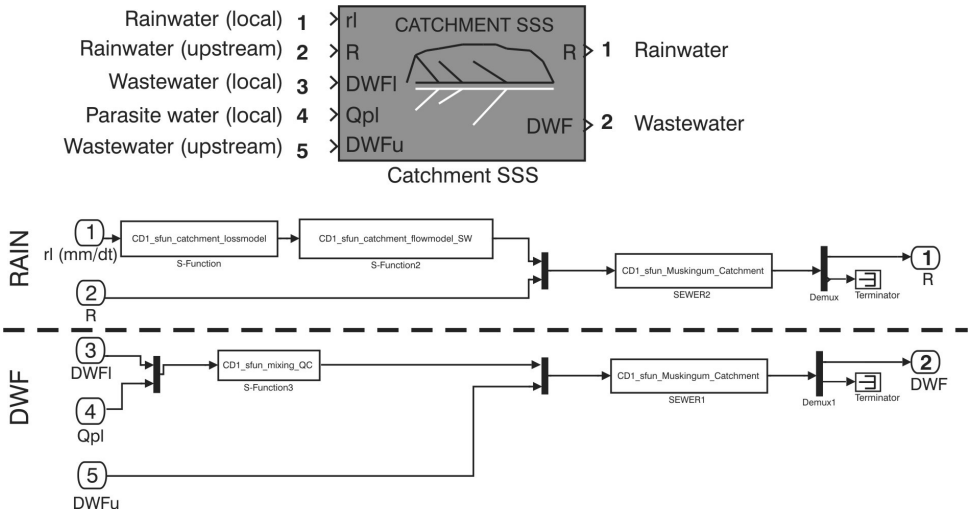
No initial conditions applied

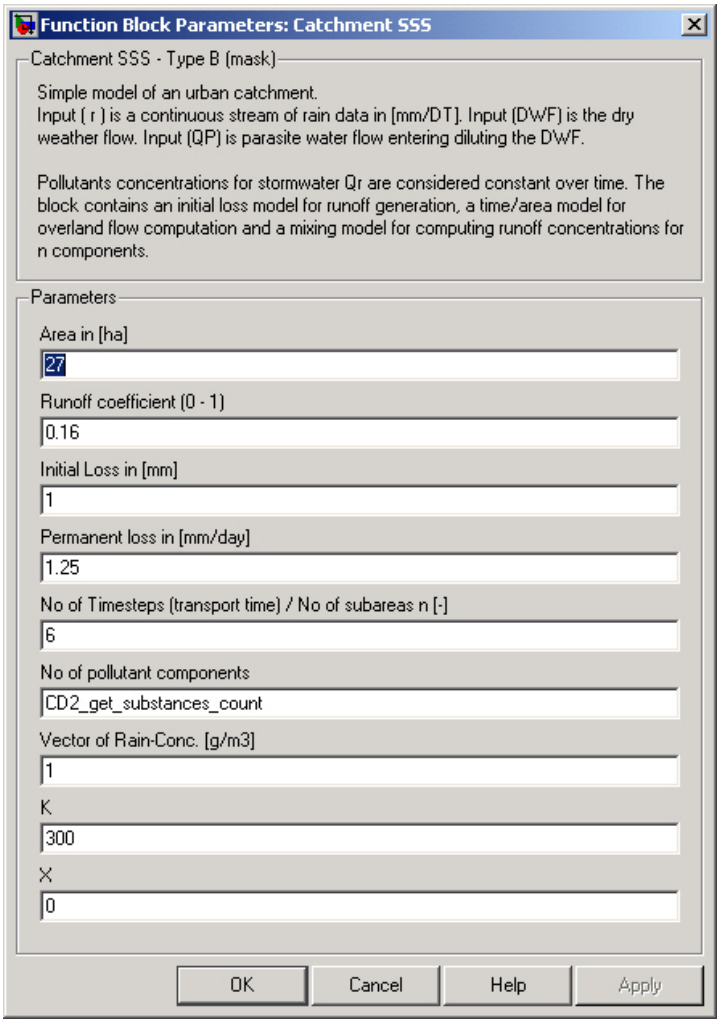
Theory:

For details of the utilized blocks/s-functions see chapters

5.4.1 Simplified Muskingum .
5.4.2 Catchment Loss Model
5.4.3 Catchment Flow Model - SW (storm water)
8.1 Mixing

5.2 Catchment SSS





Function:

Catchment SSS block is designed to simulate a separate sewer system on catchment level. Equivalent to the CSS block, it copes with the major drainage-related processes in an urban

area and returns for each time step both the current amount and pollutant concentration of the aggregated outflow from the catchment. In contrast to the CSS block, streams of Storm water runoff and wastewater are treated in separate stream, leading to two dynamic block output (R and DWF). Identically to the CSS block, associated dynamic inputs can be distinguished for inputs that (a) originate from the catchment and (b) inputs which originate from upstream and are to be routed through the catchment.

The modified scheme allows both, feeding of the uppermost block ($Q_{I,U}$) as well a distributed feeding of blocks ($Q_{I,L}$). Thus, inputs provided such as the rain intensities (*rl*) acting on the catchment, the dry weather flows generated in the catchment (DWF_L) and parasite water infiltrating into the sewer system (*Qpl*) are distributed homogeneously within the catchment. Flows from upstream of the catchment are all the way routed through and thus are fed to the uppermost sub-block. For the SSS block two ports allow the dynamic inputs to the storm and wastewater sewer (R and DWFu respectively).

Input:

r_L	Rain volume per time step [mm/ Δt]. Provided by source block *CD1_rainread*.
Ru	Dynamic dry weather flow from an upstream catchment [q [m³/s] C_1[g/m³] C_2[g/m³]
$Q_{DWF,L}$	Dynamic dry weather flow [q [m³/s] C_1[g/m³] C_2[g/m³];
$Q_{P,L}$	Dynamic flow parasite water [q [m³/s] C_1[g/m³] C_2[g/m³]
$Q_{DWF,U}$	Dynamic dry weather flow from an upstream catchment [q [m³/s] C_1[g/m³] C_2[g/m³]

Output:

Q_R	Stormwater (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³]
Q_{DWF}	Dry Weather (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³]

Parameters:

A	Catchment Area A [ha]
φ	Runoff coefficient φ [-]. In the range of φ = 0...1.
hi	Initial loss [mm]
hp	Permanent loss [mm/ day]
n_{TA}	Number of sub areas/subreaches n_{TA} [-]
n_P	Number of pollutants n_P [-]
C_{RAIN}	Vector of pollutant concentrations of storm water flow [g/m³]
K	Muskingum parameter describing the time required for a discharge wave travelling through the reach [s]. **K applies to one subreach** and does not cover travelling time for all sub reaches.
X	Dimensionless weighting factor that relates to the amount of wedge storage [-] in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).

With regard to flow dynamics, stormwater and dry weather flow is routed using equivalent catchment/model parameters n_{TA}, K and X respectively.

Initial Conditions:

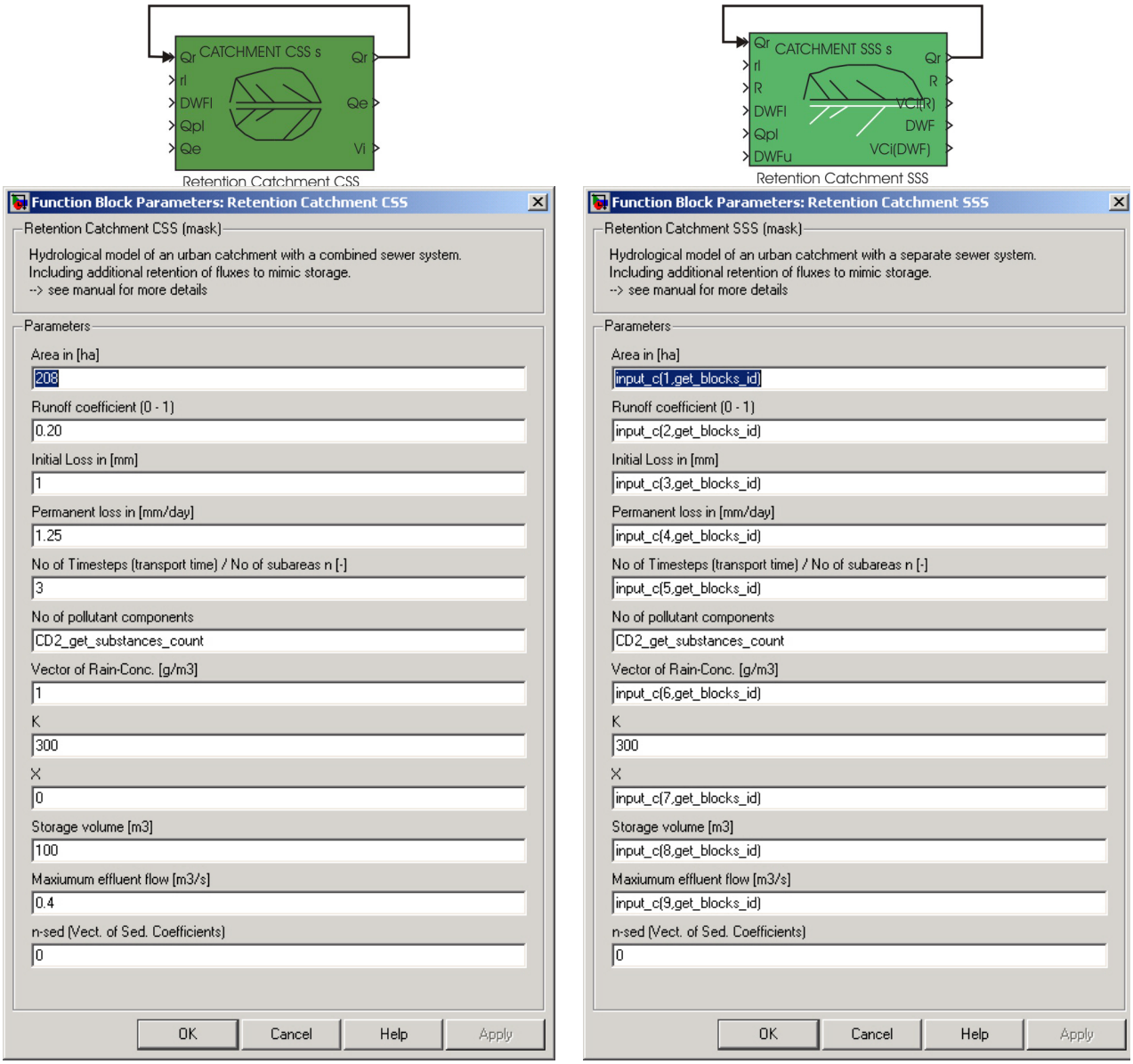
No initial conditions applied

Theory:

For details of the utilized blocks/s-functions see chapters

5.4.1	Simplified Muskingum .
5.4.2	Catchment Loss Model
5.4.3	Catchment Flow Model - SW (storm water)
8.1	Mixing

5.3 Retention Catchment (CSS and SSS)



Function:

The Retention Catchments (CSS and SSS) apply in principal the same underlying models as the classical catchments CSS and SSS. As the Muskingum method may not work sufficiently with regard to storage when it is applied for flat urban catchment, these advanced block types where generated.

In order to provide an additional storage ability within the block to mimic backpressure effects, a “classical” catchment is put in series with a CSO structure inside the blocks.

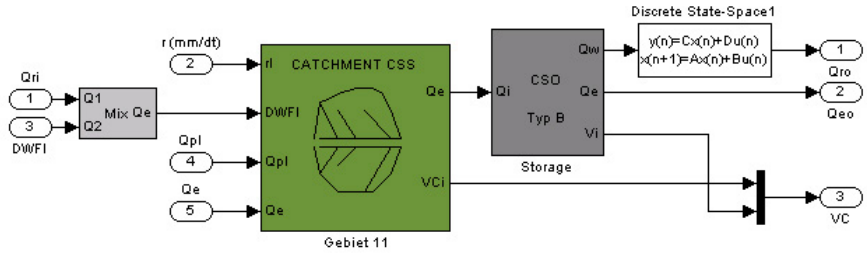


Fig. 8: Internal flow scheme of the retention type catchment CSS

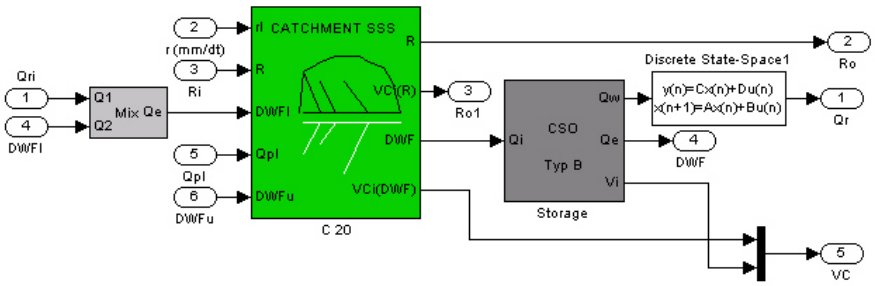


Fig. 9: Internal flow scheme of the retention type catchment SSS

The storage volume introduced by the storage volume of the virtual CSO structure is – with respect to real conditions – interpretable as part of the storage volume included in the sewer/catchment system. The throttled flow leaving the virt. CSO and consequently the catchment system is as well definable by the user. As backpressure conditions occur usually with a filled sewer system, the maximum throttled flow is interpretable as maximum flow obtainable from the sewer system under filled/pressure conditions. All flows exceeding that value are usually stored in the system, either in the sewer or – worst case – as overflow at manholes. In any case, the volume needs to be stored in the system. To mimic the behaviour numerically, flows are fed back from Qr(outflow) to Qr (inflow). The ports can be connected directly where a “discrete state space” block is included in the block after the flow Qw to avoid a numerical loop. Thus, storage of water and matter is numerically delayed by one time step.

For the case of the CSS, the full stormwater/DWF stream is recycled and introduced as additional (distributed) DWF to the catchment system.

For the case of the SSS the DWF stream exclusively is recycled. Stormwater is not subjected to an additional retention. In case fluxes occurring within the stormwater stream of a SSS are of special concern, a recycle feature can be included similarly.

Block Types:

- Catchment CSS
- Catchment SSS
- CSO Type B

Input:

see blocks

- Catchment CSS
- Catchment SSS
- CSO Type B

Additional input:

Q_{Ri} Recycled stream for additional retention.

Output:

CSS
Q_e Combined (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³]
V_{CI} Currently stored volume and concentration [V [m³] C_1[g/m³] C_2[g/m³]

SSS
Q_R Stormwater (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³]
Q_{DWF} Dry Weather (out) flow at the catchment outlet [q [m³/s] C_1[g/m³] C_2[g/m³]
V_{CI} Currently stored volume and concentration of DWF stream [V [m³] C_1[g/m³] C_2[g/m³]

Parameters:

As required for blocks

- Catchment CSS
- Catchment SSS
- CSO Type B

Initial Conditions:

No initial conditions applied

5.4 Underlying Blocks in catchment models

5.4.1 Simplified Muskingum routing method

Rainfall measurements are likely given as rain volumes, accumulated during a certain period of time (usually in the range of 5 – 10 min intervals). The rain r_i [mm] is the volume of rain fallen between t_{i-1} and t_i time. Consequently the derived flow does not describe the actual flow at time t_i but the mean flow between t_{i-1} and t_i anyway. Therefore the Muskingum equation (Roberson *et al.*, 1995b) is newly derived using a simplified numerical scheme (Achleitner *et al.*, 2006; Motiee *et al.*, 1997).

$$V = \frac{V_i + V_{i-1}}{2} = K \cdot Q_{E,i} + K \cdot X \cdot (Q_{I,i} - Q_{E,i})$$

$$\frac{\Delta V}{\Delta t} = \frac{V_i - V_{i-1}}{\Delta t} = Q_{I,i} - Q_{E,i}$$

Eliminating the stored volume V_i leads to

$$2 \cdot K\left[X \cdot Q_{I,i} + (1 - X) \cdot Q_{E,i}\right] - V_{i-1} = \left(Q_{I,i} - Q_{E,i}\right) \cdot \Delta t + V_{i-1}$$

Consequently the mean outflow $Q_{E,i}$ is

$$Q_{E,i} = C_X \cdot Q_{I,i} + C_Y \cdot V_{i-1}$$

with

$$C_X = \frac{\frac{\Delta t}{2} - K \cdot X}{\frac{\Delta t}{2} + K \cdot (1 - X)} \quad \text{and} \quad C_Y = \frac{1}{\frac{\Delta t}{2} + K \cdot (1 - X)}$$

In contrast to the original discrete scheme used, only two instead of three parameters are required. The volume stored at time t_i is

$$V_i = \left(Q_{I,i} - Q_{E,i}\right) \cdot \Delta t + V_{i-1}$$

For numerical stability, different requirements are to be fulfilled. For assessing the wave travelling through the reach, the sampling time Δt is to be smaller than the flow time K.

$$\Delta t \le K \; ; \; 1 \le \frac{K}{\Delta t} \quad \text{…Requirement (1)}$$

Further, the coefficients C_X and C_Y are required to be positive for a positive contribution of the inflow and the stored volume to the outflow.

$$C_X \ge 0; C_Y \ge 0 \quad \text{…Requirement (2)}$$

Derived from these requirements the ratio of $K/\Delta t$ is to be in the range of

$$1 \le \frac{K}{\Delta t} \le \frac{1}{2X}$$

for obtaining numerical stability and avoiding negative values in the coefficients:

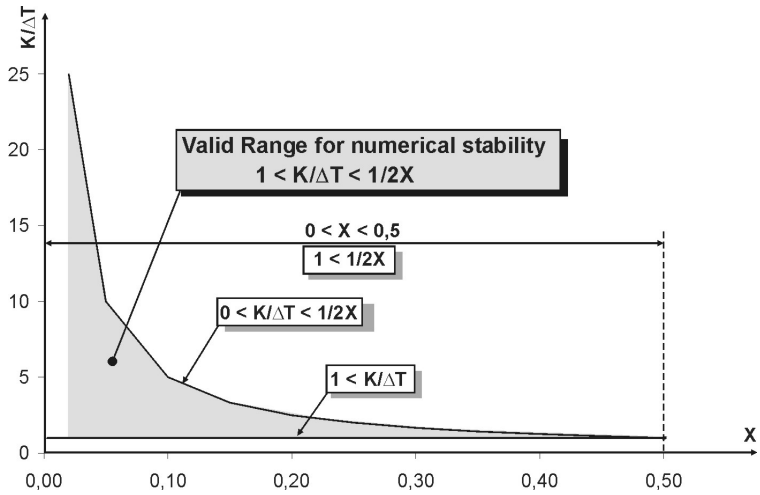


Fig. 10: Valid range for numerical stability in the simplified discrete Muskingum scheme.

Consequently the range for *X* – values is defined as well from this equation, where *X* is in the range of *0 < X < 0.5*. Only for this range the above stated relation of

$$1 \le .... \le \frac{1}{2X}$$

holds true. For dealing with *n* subreaches, the formulas are rearranged and the nomenclature is taken as followed to include the numbering of subreaches *j (1...n)*.

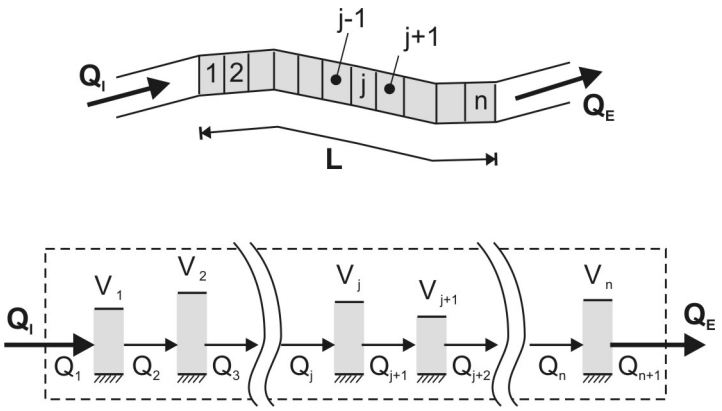


Fig. 11: Schematic on nomenclature for multiple subreaches

The Muskingum parameter *K* applies to the total reach. For simplicity a reach is always split for *n* equal subreaches, each having an associated Muskingum parameter *K'*:

$$K' = K/n$$

The overall wave travelling time *K* is substituted by *K'* being the travelling time for each subreach ($K' = K/n$). The outflow of the reach *j* is:

$$Q_i^{j+1} = C_X \cdot Q_i^j + C_Y \cdot V_{i-1}^j$$

The storage volume at time t_i in reach j is

$$V_i^j = \left(Q_i^j - Q_i^{j+1}\right) \cdot \Delta t + V_{i-1}^j$$

So far, the model only considers flow entering the most upstream compartment. This is feasible, as long as the method is used for only transport pipes without any lateral flow. For applications such as river or catchment routing, flow entering along the pathways is necessary. This is the case for dry weather flow that is generated all over the catchment but also for rainfall that is spatially distributed. A general scheme including compartment-wise inflows is illustrated in the following.

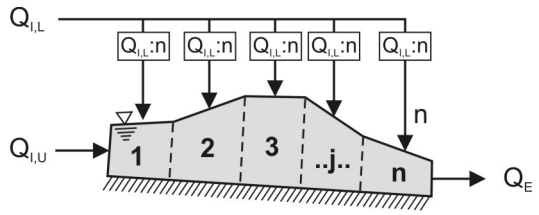


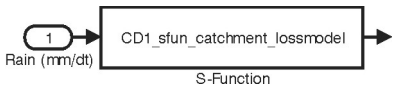
Fig. 12: Muskingum routing with multiple sub-reaches having upstream and compartment-wise inflows.

The total flow introduced "locally" $Q_{I,L}$ is – according to the area and subreaches - evenly distributed over n sub-reaches ($Q_{I,L} / n$). The formulas do not have to be changed as the total local inflow comprises both, the upstream flow and the locally generated flow.

$$Q_i^{j+1} = Q_i^j \cdot C_X + V_{i-1}^j \cdot C_Y = \left(Q_i^j + \frac{Q_{I,L}}{n}\right) \cdot C_X + V_{i-1}^j \cdot C_Y$$

described function itself has been as well extended

5.4.2 Catchment Loss Model



Function:

Applies initial and permanent loss to a given precipitation, responsible for the generation of effective runoff height.

Input:

r_R Rain volume per time step [mm/ Δt]. Provided by e.g. source block *CD1_rainread*.

Output:

h_e Effective runoff per time step [mm/ Δt].

Parameters:

S function parameters [hi , hp , φ]

hi Initial loss [mm]
hp Permanent loss [mm/ Δt]
φ Runoff coefficient φ [-]. In the range of φ = 0...1.

Initial Conditions:

No initial conditions to be applied

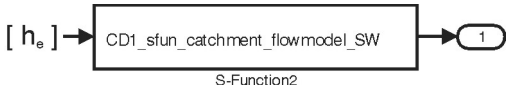
Theory:

The underlying model is a virtual basin having the volume (height) of h_I. The initial loss is therefore represented by a filling of the basin, where the effective runoff h_e is computed as spilled volume per time step Δt .

$$h_e = \left(r_R - h_I\right) \cdot \varphi \geq 0$$

Permanent loss per time step h_P is taken into account only during dry weather periods, used for emptying the virtual basin.

5.4.3 Catchment Flow Model - SW (storm water)



Function:

Block for generating outflow Q_e from a given effective runoff height h_e [mm / Δt]. Pollutant concentrations – given as constant parameters - are added to the flow.

Input:

h_e Effective runoff height per time step [mm/ Δt] at the catchment outlet.

Output:

Q_e Stormwater runoff in [m³/s] at the catchment outlet, including concentrations of substances carried. Concentrations applied are introduced as constant parameter C_{RAIN}.

Parameters:

S function parameters [A, $n_{P'}$, C_{RAIN}, t_{SAMP}]

A Catchment Area [m²].
n_P Number of pollutants carried n_P [-]
C_{RAIN} Vector of pollutant concentrations of storm water flow [g/m³]
Δt Sampling rate (time steps) in [s]. Δt is inherited from the global setting of sampling times.

Theory:

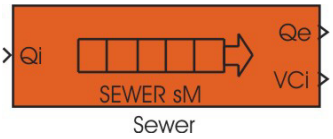
The S-function is responsible for creating a flow vector representing the storm water flow at the catchment outlet only. The function covers the transfer from dynamic rain height h_e in terms of [mm/Δt] to flow rate q_e [m³/s] using the catchment area A. The resulting outflow rate from the catchment is calculated as

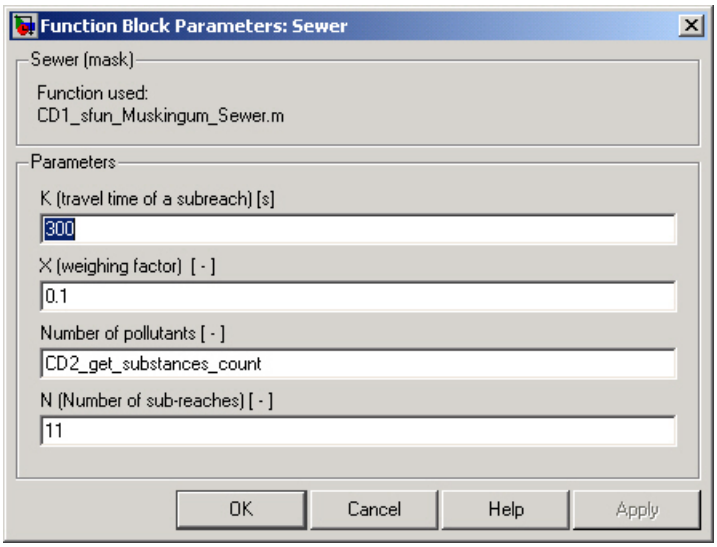
$$q_e[m^3/s] = \frac{h_e[mm/\Delta t]}{1000 \cdot \Delta t[s]} \cdot A[m^2]$$

The number of substances carried n_P must meet the length of the vector holding the pollutant concentrations (C_{RAIN}). The resulting output vector Q_e has the form of

$$Q_e = \left\lfloor q_e \quad C_{RAIN,1} \quad \cdots \quad C_{RAIN,K} \quad \cdots \quad C_{RAIN,n_P} \right\rfloor$$

5.5 Sewer





Function:

Block for flow / pollutant routing in a sewer by means of the Muskingum routing method applied for water and matter. The block may be used for sewer stretches not subjected to additional surface runoff or DWF along ist flow path. Upstream inflow is required as dynamic input Q_i.

Input:

Q_i Inflow to be routed in the sewer.

Output:

Q_e Effluent flow at the sewer outlet.
V_{Ci} Current volume and concentrations stored in the block.

Parameters:

K Muskingum parameter describing the time required for a discharge wave travelling through the reach [s]. **K applies to one subreach** and does not cover travelling time for all sub reaches.
X Dimensionless weighting factor that relates to the amount of wedge storage [-] in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).
n_P Number of pollutants n_P [-]
n_T Transport time defined as number of time steps/sub-reaches.

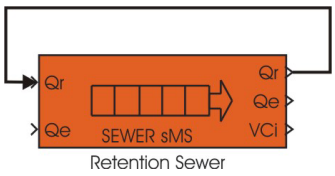
Initial Conditions:

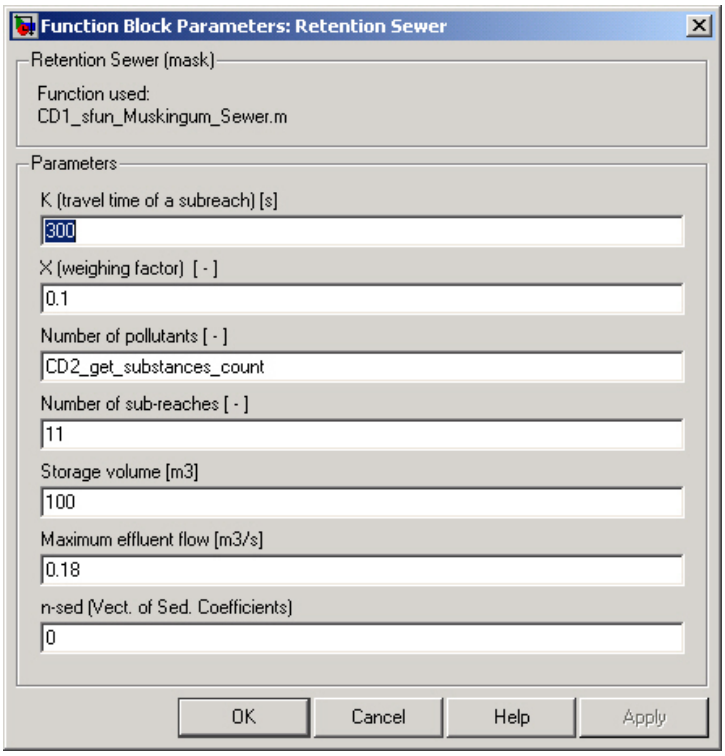
No initial conditions to be applied.

Theory:

See chapter
5.4.1 Simplified Muskingum routing method

5.6 Retention sewer





Function:

The Retention Sewer block utilizes as well the Muskingum routing model as the regular Sewer block. As the Muskingum method may not work sufficiently with regard to storage when it is applied for flat stretches, this advanced block type was generated. In order to provide a an additional storage ability within the block to mimic backpressure effects, a “classical” sewer block is put in series with a CSO structure inside this new block.

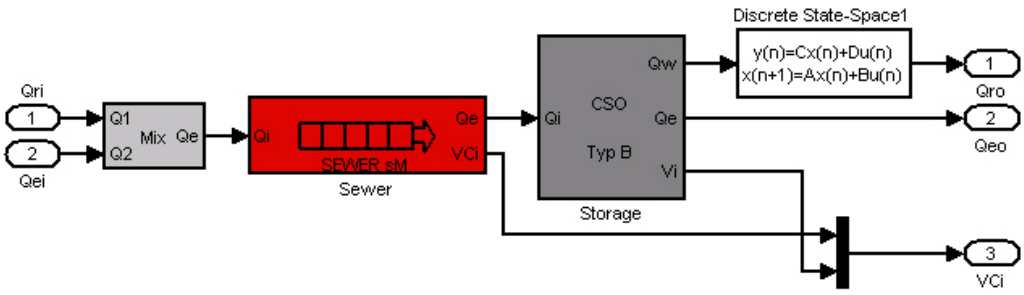


Fig. 13: Internal flow scheme of the retention type sewer block

Analoguous to the scheme applied in the catchment blocks, an additional storage volume is introduced. The virtual CSO structure is – with respect to real conditions – interpretable as part of the storage volume included in the sewer system. The throttled flow leaving the virtual CSO and consequently the catchment system is as well definable by the user. As backpressure conditions occur usually with a filled sewer system, the maximum throttled flow is interpretable as maximum flow obtainable from the sewer under filled/pressure conditions. All flows exceeding that value are usually stored in the system, either in the sewer or – worst

case – as overflow at manholes. In any case, the volume needs to be stored in the system, thus is in this model fed back using a recycle flow. To avoid a numerical loop, a “discrete state space” block is included after the flow Q_W.

Block Type:

- Sewer
- CSO Type B

Input:

Q_{ri} Inflow to be routed in the sewer.
Q_{ei} Inflow from upstream system.

Output:

Q_e Effluent flow at the sewer outlet.
Q_r Recycled storage flow.
V_{Ci} Current volume and concentrations stored in the block.

Parameters:

See blocks

- Sewer
- CSO Type B

Initial Conditions:

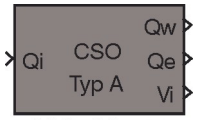
No initial conditions to be applied.

Theory:

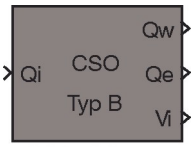
See chapters of underlying blocks

5.4.1 Simplified Muskingum routing method
5.7 CSO (Types A and B)

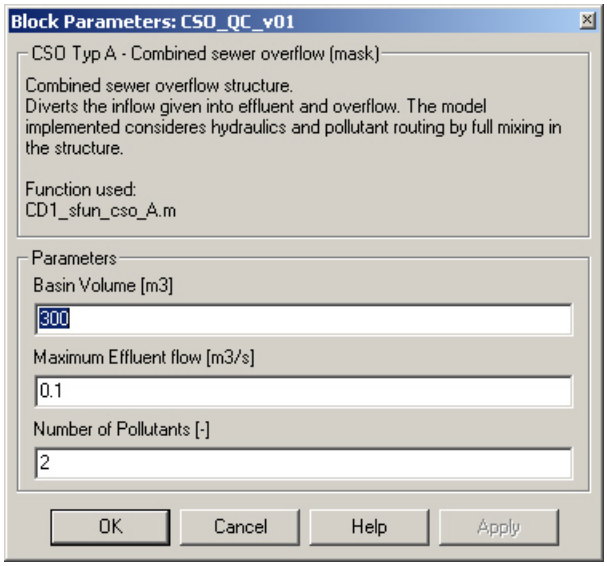
5.7 CSO (Types A and B)

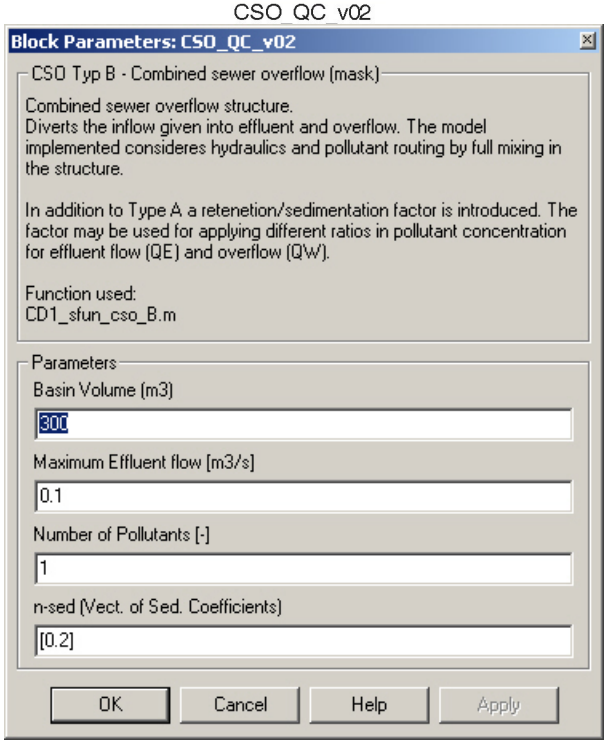


CSO_QC_v01



CSO_QC_v02





Function:

Simulation of an overflow structure for either combined or separate sewer. Inflow is routed downstream via the effluent outlet (Q_E) which is limited by a maximum effluent flow rate $Q_{E,MAX}$ [m³/s]. Flow exceeding the structures storage capacity (V_{MAX}) is routed via the CSO overflow (Q_W [m³/s]). With the model implemented hydraulic and pollutants routing is based on instant and ideal mixing in the CSO structure. CSO Type B contains, in contrast to Type A, a simplified modelling option of sedimentation. Using a linear sedimentation coefficient η_{sed} in CSO type B allows modelling of lower pollutant concentrations present in the overflow (Q_W) than in the effluent flow (Q_W).

Input:

Q_i Inflow to CSO structure.

Output:

Q_e Effluent flow at the CSO structure.
Q_w Overflow generated when CSO storage volume is exceeded.

Parameters:

CSO Type A [V_{MAX}, $Q_{e,MAX}$, $n_{P'}$, Δt]
CSO Type B [V_{MAX}, $Q_{e,MAX}$, $n_{P'}$, η_{sed}, Δt]

V_{MAX} Basin volume (storage volume) [m³]
$Q_{E,MAX}$ Maximum Effluent flow [m³/s]
n_P Number of pollutants carried [-].
η_{sed} Sedimentation coefficient

Δt Sampling rate Δt [s] is inherited from global setting of simulation parameters.

Initial Conditions:

No initial conditions to be applied. The structure is considered being empty at the beginning of the simulations.

Theory:

Hydraulics of the CSO structure:

The basic differential equation of the hydraulic mass balance for the CSO structure is written as

$$\frac{\partial V}{\partial t} = Q_I(t) - Q_E(t) - Q_W(t)$$

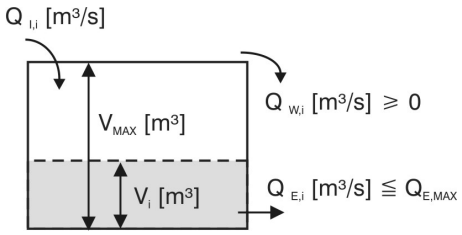


Fig. 14: Variable definitions of discrete CSO model

The flows Q_I , Q_E and Q_W are considered as mean flows occurring during the discrete timely period. Herein the volume is related to discrete points of time. The mass balance equation therefore is

$$\frac{V_i - V_{i-1}}{\Delta t} = Q_{I,i} - Q_{E,i} - Q_{W,i}$$

where i denotes the time step considered. Restrictions are usually given by the maximum Volume of the CSO structure (V_{MAX}) and the maximum outflow from the structure $Q_{E,MAX}$.

Depending on the magnitude of inflow $Q_{I,i}$ and the previous Volume stored (V_{i-1}), three different cases apply. The cases can be differentiated considering the hydraulic mass balance with no overflow $Q_W = 0$ and fully developed outflow $Q_E = Q_{E,MAX}$. The virtual volume V_i' from this mass balance denotes

$$V_i' = \left(Q_{I,i} - Q_{E,MAX}\right) \cdot \Delta t + V_{i-1} .$$

Three cases can be distinguished:

Case No.		
1	$V_i' < 0$	Qw = 0 … No overflow $Q_E < Q_{E,MAX}$ …Outflow Q_E not fully developed
2	$V_i' > V_{MAX}$	$Q_W > 0$ …Overflow Q_W developed $Q_E = Q_{E,MAX}$ …Outflow Q_E fully developed
3	$0 < V_i' < V_{MAX}$	Qw = 0 …No overflow $Q_E = Q_{E,MAX}$ …Outflow Q_E fully developed

In case V_i' falls below zero (case 1), the maximum outflow $Q_{E,MAX}$ from CSO structure did not develop throughout the whole time and the structure falls dry. In order to keep the hydraulic mass balance, the outflow rate is adjusted (reduced).
In case of a positive V_i' (cases 2 and 3) the maximum flow rate $Q_{E,MAX}$ develops. An overflow occurs when having a virtual volume large than the maximum ($V_i' > V_{MAX}$).

For calculating the unknowns V_i , $Q_{I,i}$ and $Q_{W,i}$ the following equations apply for the respective case:

Case 1

$$V_i = 0$$

$$Q_{E,i} = \frac{V_{i-1}}{\Delta t} + Q_{I,i} \le Q_{E,MAX}; \quad Q_{W,i} = 0$$

Case 2

$$V_i = V_{MAX}$$

$$Q_{E,i} = Q_{E,MAX}; \quad Q_{W,i} = Q_{I,i} - Q_{E,MAX} - \frac{V_{MAX} - V_{i-1}}{\Delta t}$$

Case 3

$$V_i = \left(Q_{I,i} - Q_{E,MAX}\right) \cdot \Delta t + V_{i-1}$$

$$Q_{E,i} = Q_{E,MAX}; \quad Q_{W,i} = 0$$

Mixing and Settling

The here presented scheme builds on the simplified hydraulic scheme for CSO storage. Flows are taken into account that represent mean flows occurring during a time step Δt. Similar, the concentrations entering or leaving the chamber are mean concentrations over a time step.

Flow entering the basin is assumed to be fully mixed with the volume in the structure. The new concentrations being present in the chamber are then taken for the effluent and overflow concentrations. This is done for (a) simplification of calculations and (b) in order to avoid numerical shortcomings (e.g. negative concentrations).

Hydraulic parameters/variables

$Q_{I,i}$	Inflow for timestep i
V_{i-1}	Volume stored for the previous time step
$V_{QI,i}$	Volume added to the storage by inflow $Q_{I,i}$ during time step Δt
V'_i	Volume stored for the current time step (including overflow volume)
V_i	Volume stored for the current time step
$Q_{W,i}$	Overflow for timestep i
$Q_{E,i}$	Outflow for timestep i

Substance parameters/variables

C_{QIi}	Concentration in the inflow
$C_{V,i-1}$	Concentration of stored volume in previous time step i-1
$C'_{V,i}$	Concentration of stored volume in current time step i related to the total volume V'_i
$C_{V,i}$	Concentration of stored volume in current time step i related to the stored volume V_i
$C_{QW,i}$	Concentration of overflow volume in current time step i
$C_{QE,i}$	Concentration of outflow volume in current time step i

Mass balancing is done by first mixing the inflow QI with the stored volume of the previous time step.

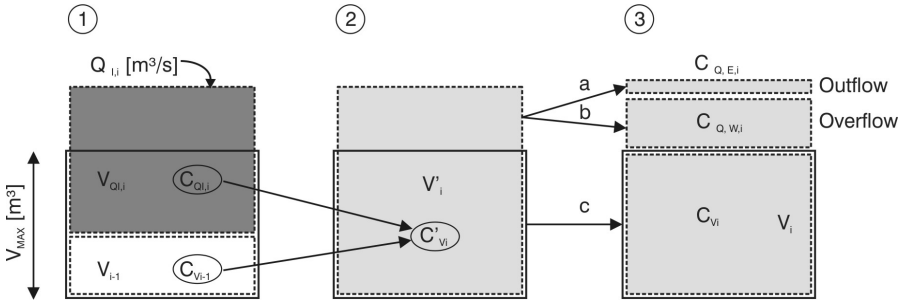


Fig. 15: Scheme for flow and pollutant mass balance

$$V'_i = Q_{I,i} \cdot \Delta t + V_{i-1}$$

The concentration in this volume is

$$C'_{Vi} = \frac{C_{QI,i} \cdot Q_{I,i} \cdot \Delta t + C_{V,i-1} \cdot V_{i-1}}{V'_i} \qquad \text{for} \qquad V'_i > 0$$

In order to account for settling of matter, the concentrations in the overflow volume are reduced using a settling coefficient η_{SED} to virtually account for the sedimentation. The concentrations present in the overflow are reduced by:

$$C_{QW,i} = C'_{V,i} \cdot (1 - \eta_{SED})$$

The mass balance for the substance matter present in the volume *V'i* leads to an increased concentration in the remaining volume V_i . Considering the effluent concentration $C_{QE,i}$ being the same as the chamber concentration $C_{V,i}$ leads to

$$C_{V,i} = \frac{C_{QI,i} \cdot Q_{I,i} \cdot \Delta t + C_{V,i-1} \cdot V_{i-1}}{Q_{E,i} \cdot \Delta t + V_i} \cdot \left(1 - \frac{Q_{W,i} \cdot \Delta t}{Q_{I,i} \cdot \Delta t + V_{i-1}} \cdot (1 - \eta_{SED}) \right)$$

$$C_{QE,i} = C_{V,i}$$

The concentration in the overflow is therefore

$$C_{QW,i} = \frac{C_{QI,i} \cdot Q_{I,i} \cdot \Delta t + C_{V,i-1} \cdot V_{i-1}}{Q_{I,i} \cdot \Delta t + V_{i-1}} \cdot \left(1 - \eta_{SED}\right).$$

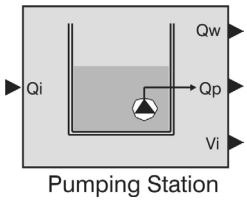
For numerical stability, following terms are required to be > 0:

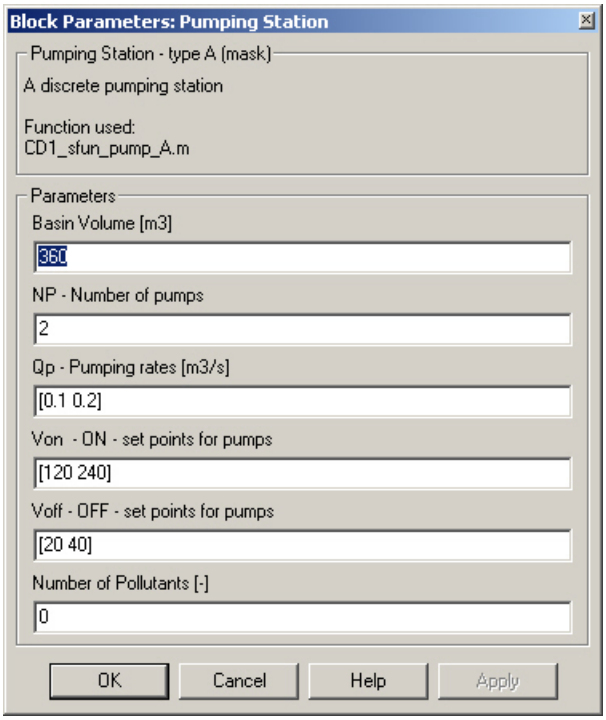
$$Q_{I,i} \cdot \Delta t + V_{i-1} > 0 \text{ and } Q_{E,i} \cdot \Delta t + V_i > 0$$

Physical interpretation of the first term would be that no inflow occurs ($Q_{I,i} = 0$) and the basin was empty in the last time step ($V_{i-1} = 0$). Therefore, the concentration in the chamber and the effluent concentration are set to zero ($C_{E,I} = C_{V,I} = 0$). Setting the concentration to zero is done for the overflow quality as well ($C_{QW,i} = = 0$) , since hydraulically no overflow is given. The second boundary condition fails when no outflow occurs ($Q_{E,i} = 0$) and if the basin is

empty at the present time step ($V_i = 0$). Again the concentrations are set to zero ($C_{QW,i} = C_{QE,i} = C_{V,i} = 0$), due to having an empty basin at the present time step, having not outflow and overflow.

5.8 Pumping station





Function:

The implemented concept of a simplified discrete formulation of a pumping station, is – equivalent to other models used - based on mean flows from and to the storage structure. The central idea is to describe a pumping station with a number of pumps, each with it's fixed pumping rate ($Q_{P,k}$) and it's set points (water levels) for turning pump either on ($h_{P,k}^{ON}$) or off ($h_{P,k}^{OFF}$).

The number of installed pumps including their characteristics is up to the user and adaptable.

Input:

Q_i Inflow to CSO structure.

Output:

Q_p Pumped flow as a sum of all pumps installed [q [m³/s] C_1[g/m³] C_2[g/m³].
Q_w Overflow generated in excess of the pumped flow [q [m³/s] C_1[g/m³] C_2[g/m³].
V_i Current volume and concentrations stored in the pumping station [V[m³] C_1[g/m³] C_2[g/m³].

Parameters:

V_p Maximum storage volume of pumping station [m^3].
NP Number of installed pumps [-].
Q_p Vector of pumping rates for all pumps installed [m^3/s].
V_{ON} Vector of volume representing the ON-set points for the pumps [m^3].
V_{OFF} Vector of volume representing the OFF-set points for the pumps [m^3].
n_P Number of pollutants [-]

Initial Conditions:

No initial conditions to be applied. The structure is considered being empty at the beginning of the simulations.

Theory:

The storage volume is defined by the basin volume V_{MAX} of the structure.

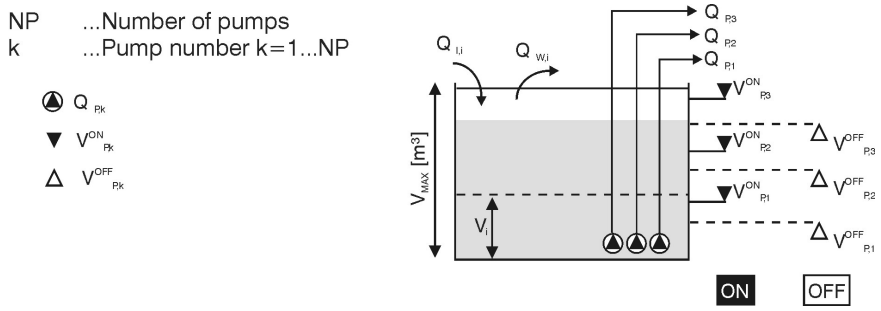


Fig. 16: Schematic of pumping station with multiple (here 3) pumps

By definition the pumping rates are defined as

$$Q_P = [\quad Q_{P,1} \quad .... \quad Q_{P,k} \quad .... \quad Q_{P,NP} \quad]$$

Set points (water levels) are

$$h_P^{ON} = [\quad h_{P,1}^{ON} \quad .... \quad h_{P,k}^{ON} \quad .... \quad h_{P,NP}^{ON} \quad]$$

$$h_P^{OFF} = [\quad h_{P,1}^{OFF} \quad .... \quad h_{P,k}^{OFF} \quad .... \quad h_{P,NP}^{OFF} \quad]$$

Alternatively these may be written in terms of volume using the base area A of the structure

$$0 \le V_P^{ON} = A \cdot h_P^{ON} \le V_{MAX}$$

$$0 \le V_P^{OFF} = A \cdot h_P^{OFF} \le V_{MAX}$$

where set points are to be within the range of the structure storage capacity V_{MAX}. Requirement for the set points is that the ON points are more elevated than the corresponding OFF points.

$$h_{P,k}^{OFF} \le h_{P,k}^{ON}$$

Further the set points are to be monotonically increasing in elevation.

$$\begin{aligned} h_{P,1}^{ON} \quad &\le \quad h_{P,k}^{ON} \quad \le \quad h_{P,NP}^{ON} \\ h_{P,1}^{OFF} \quad &\le \quad h_{P,k}^{OFF} \quad \le \quad h_{P,NP}^{OFF} \end{aligned}$$

The scheme assumes the inflow and outflows as mean values over the last time step. The inflow $Q_{I,i+1}$ is further fully mixed with the structures content from the last time step V_i resulting in a virtual content V'_{i+1}.

$$V'_{i+1} = V_i + Q_{I,i+1} \cdot \Delta t$$

This virtual content represents the volume which would have been accumulated without pumping. Operation of pumps is driven by the water level (or by the stored volume respectively) where the order in which pumps are operating is according to the ON/OFF set

points in increasing sequence. Pump 1 is operating in case the virtual volume V'_{i+1} tops the ON set point of the pump.

$$V'_{i+1} = V^1_{i+1} \geq V^{ON}_{P,1}$$

For homogeneous notation for all pumps, the virtual volume V'_{i+1} is termed V^1_{i+1} since it is compared with the set point volume of the first pump. In case there is no operation of the pump the pumping rate for considered time step ($Q'_{P,1}$) is set to zero.

$$Q'_{P,1} = 0$$

For the case that the pump is operating it must be checked whether the volume available is sufficient for operating the pump throughout the whole time span Δt. Secondly, the remaining volume must be larger than the OFF set point of the pump ($V^{OFF}_{P,1}$).

$$V^1_{i+1} - Q_{P,1} \cdot \Delta t \geq V^{OFF}_{P,1} \geq 0$$

Since the OFF set point is required to be greater than zero, both requirements are fulfilled simultaneously. For the case that sufficient volume is available for full operation, the pumping rate for the considered time step is set to the given pumping rate.

$$Q'_{P,1} = Q_{P,1}$$

If operation is not possible for the full period Δt, the pumping rate $Q'_{P,1}$ is reduce instead of partial operation with the original pumping rate $Q_{P,1}$.

$$Q'_{P,1} = \left(V^1_{i+1} - V^{OFF}_{P,1}\right) / \Delta t$$

This does not reflect the real operation but is necessary for fitting in the discrete time scheme (time steps Δt) used. Finally the mass balance, respectively the volume withdrawn (V^1_{i+1} - $V^{OFF}_{P,1}$) remains the same. This procedure is redone for each of the remaining pumps starting at a reduced volume.

$$V^2_{i+1} = V^1_{i+1} - Q'_{P,1} \cdot \Delta t$$

The procedure using a generalized formulation for processing all available pumps is shown in the following figure, wherein the procedure is looped for the number of pumps *(NP)* used.

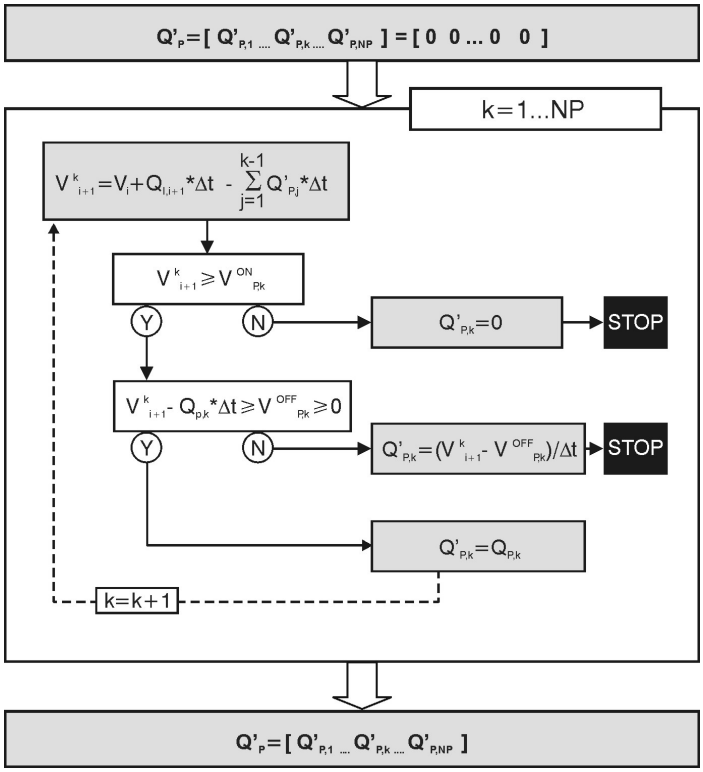


Fig. 17: Numerical scheme for a pumping station based on discrete formulation

The resulting vector of pumping rates defines the total volume withdrawn

$$Q'_P = [\quad Q'_{P,1} \quad \cdots \quad Q'_{P,k} \quad \cdots \quad Q'_{P,NP} \quad]$$

$$V'_P = \sum_{k=1}^{NP} Q'_{P,k} \cdot \Delta t$$

The remaining volume in the storage tank denotes therefore as

$$V_{i+1} = V'_{i+1} - V'_P = V_i + Q_{I,i+1} \cdot \Delta t - V'_P$$

Still this volume possibly exceeds the storage volume V_{MAX}. of the structure.

$$V_{i+1} \geq V_{MAX}$$

If this is the case, the exceeding volume is withdrawn via an overflow weir.

$$Q_W = \left(V_{i+1} - V_{MAX}\right) / \Delta t$$

Consequently the remaining volume in the tank equals the maximum storage capacity.

$$V_{i+1} = V_{MAX}$$

Fig. 18 shows an example for two pumps operating in parallel. No overflow is generated in the presented example.

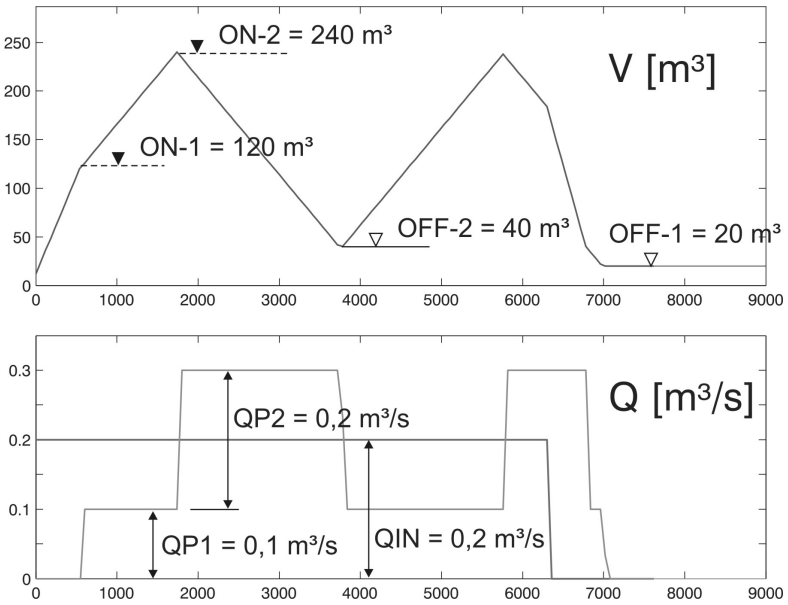
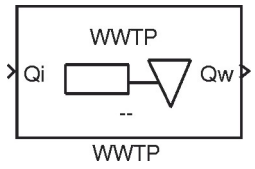
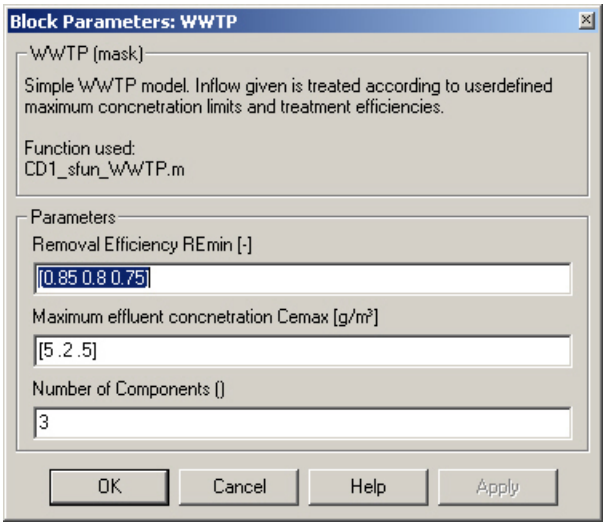


Fig. 18: Example pumping period with 2 pumps operating; (top) volume stored and (bottom) in- and outflows from the pumping station

6 WASTEWATER TREATMENT

6.1 WWTP (simple)





Function:

Models a WWTP that is consideration to fulfill predefined requirements of

- removal efficiencies $R_{E,MIN}$ [-] and
- maximum effluent concentrations $C_{E,MAX}$ [g/m³].

Thus emission standards are considered to be fulfilled, regardless the hydraulic or pollutant load associated. In the model no process are considered.

Input:

Q_i Inflow to WWTP.

Output:

Q_w Outflow from the WWTP.

Parameters:

S function parameters: [$n_{P'}$, $C_{E,MAX}$, $R_{E,MIN}$]

$R_{E,MIN}$ Required removal efficiencies [-]
$C_{E,MAX}$ Maximum effluent concentration [g/m³]
n_P Number of pollutants carried [-].

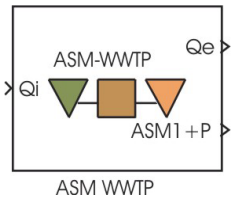
Theory:

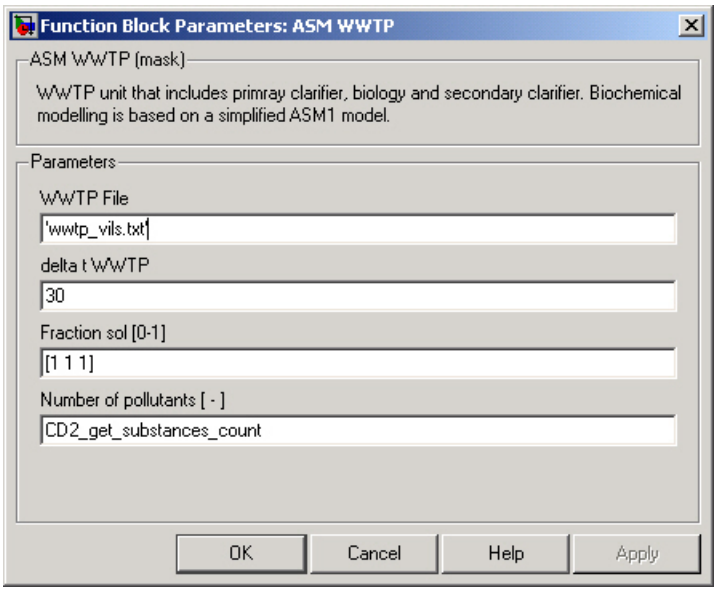
Based on user defined removal efficiencies $R_{E,MIN}$ [-] and maximum effluent concentrations $C_{E,MAX}$ [g/m³], concentrations of the inflow are reduced and assigned to the effluent flow. For each substance the more stringent of the defined prerequisites is applied.

$$C_{W,i} = \min\left[C_{E,i,MAX} \quad ; \quad C_{I,i} \cdot R_{E,MIN}\right]$$

Thus, it is assumed that the WWTP meets the prerequisites defined, regardless the magnitude of either hydraulic or pollutant loading applied.

6.2 ASM WWTP





Function:

The ASM-WWTP is designed to consider biological treatment as well as primary and secondary clarification. In contrast to other blocks, specific substances are required for usage of this model, as a modified ASM1 (Activated sludge model) is used.

Input:

Q_i Flow vector with [Q C1 C2…Cn] in the order of the global substance naming (CD_Parameters)

For the blocks input, the flow vectors is required to contain the following five substances/information as a minimum requirement to be run properly.

Temp	Inflow temperatur to the treatment plant[°C]
CODsol	COD soluble [mg/l]
CODpar	COD particulate [mg/l]
Ntot	total Nitrogen [mg/l]
Ptot	total Phosphorus [mg/l]

The respective ASM1 components are derived from these by inflow conversion of substances. Names used in the global CD_parameter setting must include these substance names as written here (case sensitive). The respective numerical values from the inflow are then used automatically within the treatment processes.

Output:

Q_e Treated outflow from the WWTP providing the flow q and substances according to the inflow to the treatment plant.

ASM1+P This output port is specific for the WWTP, returning a vector representing Qe using internal used ASM1 type substances. The vector contains the flow q [m³/s] followed by the components of the WWTP's biokinetic model in the order as shown in Tab. 2. Thus, the vector is of 11 components and does not match the global naming system used. It is rather an additional information for the user, not ment to fed to any other block.

Parameters:

WWTPfile	Definition file of the treatment plant ('file name')
ΔT_{WWTP}	Internal (constant) time step used for calculation of hydraulic and biochemical processes in the WWTP. [s]
F_{SOL}	Fraction of soluable part of the [-]
n_P	Number of pollutants [-]

Substances other than required by the WWTP are as well accepted by the block but treated without a biological conversion. Commonly substances that are surplus of the minimum required ones are bypassed and not used in the process. As a consequence this may lead to temporal differences in substance flow and mass balances between bypassed and non-bypassed matter. Thus, it was decided to avoid bypass substance and subject all substances to flow, mixing and settling processes within the treatment plant but exclude the biological conversion for the surplus of substances.

The parameter Fsol defines the soluable fraction of surplus substances. The fractions in the vector are to be entered in the order of the appearance of the surplus substances in the global substance list defined in CD_parameters.

Example:

Global substance list:	'Temp CODsol Cu CODpar Cd Ntot Ptot Bp'
and Flow vector:	$Q = [q\ C_{Temp}\ C_{CODsol}\ C_{Cu}\ C_{CODpar}\ C_{Cd}\ C_{Ntot}\ C_{Ptot}\ C_{Bp}]$
Vect. of wwtp subst.:	$Q = [q\ C_{Temp}\ C_{CODsol}\ C_{CODpar}\ C_{Ntot}\ C_{Ptot}]$
Vect. of surplus subst.:	$Q' = [C_{Cu}\ C_{Cd}\ C_{Bp}]$
Sol. fractions	$Fsol = [F_{CuSOL}\ F_{CdSOL}\ F_{BpSOL}]$

Treatment plant file:

Although the treatment plant is realized in a single block as a unit process, flexibility in the arrangement of compartments is still given. The treatment plant outline is load via a definition file that is stated in the blocks mask. Next to compartemt specific parameters such as compartment type, volume etc.., flow quantities and source are defined for each compartment. The file is ASCII code and can be opened and with every text editor.

As an example, a conventional treatment plant having a primary clarifier, anaerob and aerob reactor and a secondary settler is outlined below. The coding of the plant design is as followed

```
W r=4
P V=50 0=1
R V=300 1=1 4R=1 3=2 O2=0.0
R V=500 2=4 O2=2.0
S A=100 h=3.5 n=10 3=2
E 4E=1
```

where the different compartments are outlined together with the coded lines:

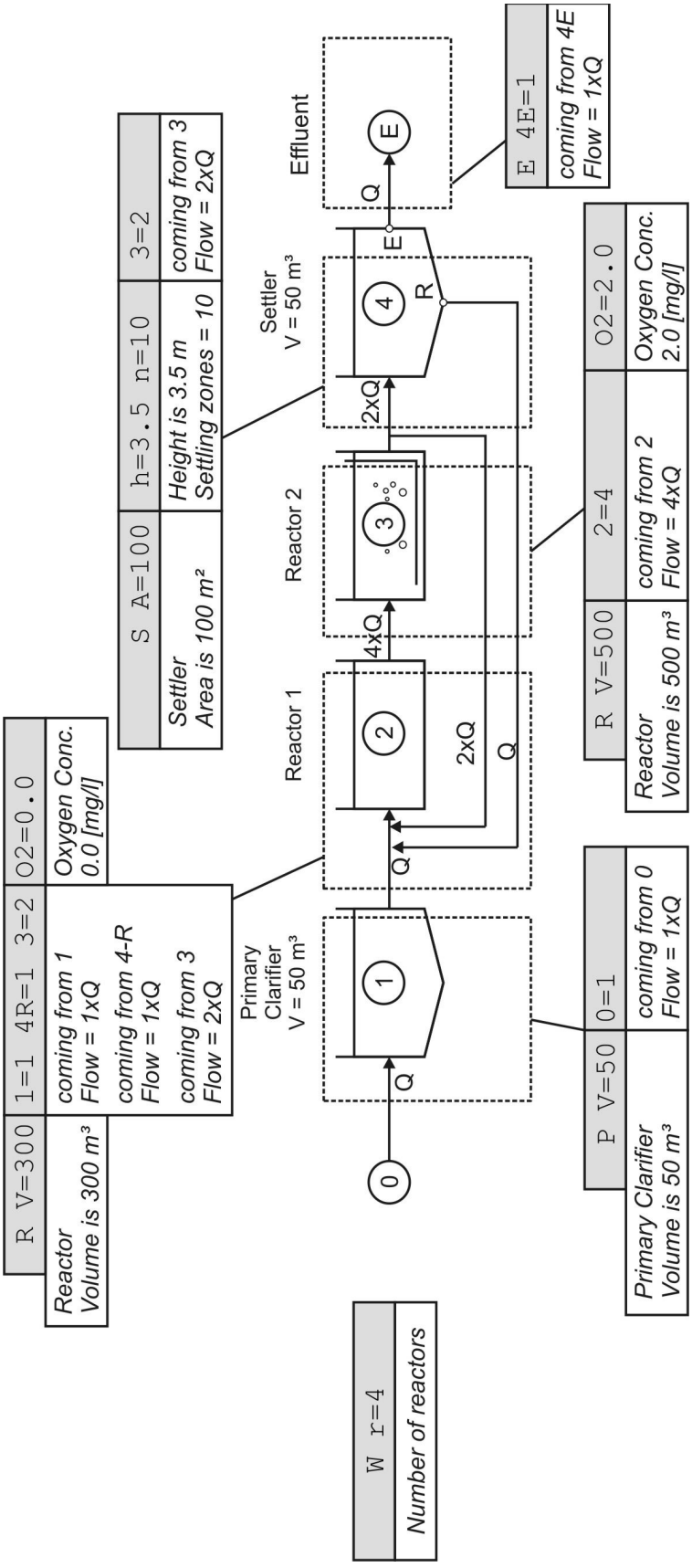


Fig. 19: Example plant layout including code explanations

The letter in each line marks the type of information followed by a blank and the parameter list.
Following the above given example, the first line (marked with W) defines the number of reactors given in the plant (e.g. *r=4*). As virtual compartments the inflow is termed as compartment “0” and the effluent is termed compartment “E”.

The following lines start with the respective identifier for the different types of compartments.

P Primary clarifier
R Reactor (with or without aeration)
S Secondary clarifier (Takacs settler)

followed by flow inputs to the compartment and compartment specific parameters. The flow quantity between the compartments is thereby defined as a multiple of the inflow. The internal numbering of compartments used to define the source of flow is made in the order of appearance of entries in the text file.

Thus, for each compartment the flow(s) entering are described by the following syntax:

$COMP\#_{SOURCE}$ =n

$COMP\#_{SOURCE}$ Number of the source compartment
n Flow quantity relative to the inflow quantity Q_{IN}

Example:

4=1 Flow $Q=1xQ_{IN}$ originating from compartment #4

Within the last row, flow from the treatment plant is defined. Effluent is thereby described by the key “E” as a virtual effluent compartment. The parameters required at each type of compartment are given below:

W - WWTP	r=total number of reactors in the WWTP
P - Primary Clarifier	V=Volume [m³]
R - Reactor	V = Volume [m³] O2 = Oxygen Concentration [mg/l]
S - Settler (Tacacs)	A=Surface area of the settler [m²] h=Height of the settler [m] n=Number of settling layer (according to Takacs)
E - Effluent	Flow from the WWTP

In the above given example the compartment 2 has inflow from 3 source. The first one originates from the compartment 1 at a rate of $1xQ_{IN}$.

Settler have – in contrast to the other types of compartment – two effluent stream. The actual effluent stream (E) is carrying the treat water and where the return sludge stream (R) is carrying activated sludge. Thus, when specifying a settler compartment as source it is required to provide the compartment number followed by the effluent type (E or R). The respective entry in the example given is 4R=1, specifying that a return sludge flow of one time the inflow Q_{IN} is diverted from the settler to the compartment 2.

For the reactors R, which are assumed to be fully mixed, the effluent stream are separated according to quantities and no distinction regarding concentration is the case. For the primary

clarifier only fluxes withdrawn from the top zone are available to be further used. Settling of matter is made, where the settled material is not proved as a separate stream.

Withdrawn quantities from the different compartment are generated automatically according to the inflow defined.

Theory:

A process description for the wastewater treatment process incorporating an ASM 1 type process model was developed. Continuous simulation of wastewater treatment plant behaviour under both dry and wet weather conditions, is based on the assumptions of the IAWQ Activated sludge model (Henze *et al.*, 1987). Further various other sub-models are included that allow simulating a complete treatment plant. The implementation follows closely the model/software denoted RUMBA. The software was developed by Rauch (1997) and is used as sub-element for other works (Harremoës and Rauch, 1996; Rauch and Harremoës, 1996, 1997). The used model is based the assumptions of the IAWQ model of Henze *et al.* (1987). Main features of the implemented model are described in the following.

The basic concept implemented is that of a recirculation plant with a primary clarifier in front followed by 2 biological reactors and a final clarifier. The oxygen set points in the two biological reactors can be determined in the setup program, so nitrification - denitrification schemes can be simulated easily. Note that the O_2 set points are not dynamic values but fixed for the simulation. Furthermore there is both an internal recycle and a recycle from the settler to the first reactor. The effluent from the primary clarifier can be directed to both reactors, i.e. the first one can be partially bypassed to model step feeding.

The following 4 quality descriptors characterize the influent to the treatment plant:

- COD soluble
- COD particulate
- total Nitrogen
- total Phosphorus

These quality descriptors are defined in the sewer system model. However, the values must be in the usual range for municipal wastewater and rain runoff. In the following an overview on the conversion parameters (including default values) used to transfer inflow quality descriptors to ASM fractions:

Tab. 1: Conversion model parameters

Symbol	Description	Default value	Sum
fs_{SI}	Fraction of S_I in CODsol	0.125	
fs_{SS}	Fraction of S_S in CODsol	0.375	1.0
fs_{XS}	Fraction of X_S in CODsol	0.500	
fp_{XS}	Fraction of X_S in CODpar	0.420	
fp_{XH}	Fraction of X_H in CODpar	0.330	1.0
fp_{XI}	Fraction of X_I in CODpar	0.250	
i_X	Fraction of N in org. matter	0.070	
i_{Xi}	Fraction of N in inert m.	0.020	
i_P	Fraction of P in X	0.01	

In the conversion model the COD components are determined more or less directly from the organic matter fraction parameters *f*. The amount of X_A in the influent is set to zero.

Primary Clarifier model

Primary clarifiers are widely used in activated sludge systems for the purpose of sedimentation and removal of suspended particulate matter in the wastewater.

The model implemented here assumes the primary clarifier to be a fully mixed tank without any conversion processes taking place but with settling and removal of particulate components. The "settling and removal model" implemented is based on an empirical relationship for the removal of particulate components as a function of the hydraulic retention time (Rauch and Harremoës, 1996; Schilling and Hartwig, 1988). The fraction of particulate matter that is not settled (X'_{IN}) is written as

$$X'_{IN} = X_{IN} \cdot (1 - \eta)$$

with the removal efficiency η for particulate matter written as

$$X'_{IN} = \eta_{\max} - (\eta_{\max} - \eta_{\min}) \cdot e^{-\eta_c \cdot V / Q_{IN}}$$

with

η …Removal efficiency for particulate matter [-]
η_{max} …maximum η
η_{min} …minimum η
η_C …Coefficient for taking the hydraulic retention time into account [T^{-1}],[1/s]
V …Primary clarifier volume [m^3]
Q_{IN} …plant influent [m^3/s]
X_{IN} …particulate matter in the plant inflow [g/m^3]
X'_{IN} …reduced particulate matter after settling (non-settled fraction) [g/m^3]

The removed amount of suspended solids is calculated directly as a fraction of the inflow. This method for describing the settling process has the advantage to take the limited amount of settable matter into account directly.

Biokinetic model

During the last three decades a large number of models of the biological processes in activated sludge treatment plants have been developed. A modified version of the IAWQ model No.1 is used for modelling biodegradation of organic material as well as biological nitrogen removal. The components implemented are shown in Tab. 2.

Tab. 2: Components used in the WWTP's biokinetic model

1	T	Temperature	Celsius
2	S_I	Inert soluble organic matter	gCOD/m^3
3	S_S	readily biodegradable organic matter	gCOD/m^3
4	S_{NO}	nitrate (NO_3^-)	gN/m^3
5	S_{NH}	ammonia (NH_4^+ + NH_3)	gN/m^3
6	S_P	Inorganic soluble phosphorus (PO_4)	gP/m^3
7	X_H	heterotrophic biomass	gCOD/m^3
8	X_A	autotrophic biomass	gCOD/m^3
9	X_S	slowly biodegradable organic matter	gCOD/m^3
10	X_I	Inert particulate organic matter	gCOD/m^3

Dissolved oxygen not introduced as component
One of the main differences to the full ASM1 model is that dissolved oxygen in the water phase is not treated as a component (i.e. state variable of the model) but as a boundary condition. The dissolved oxygen concentration S_O of both biological reactors is therefore specified in the plant outline and is not assumed to change during the simulation. Still, the given S_O values influence the process rates thus allowing to distinguish between aerobic and anaerobic conditions. Stoichiometry and process rates implemented are shown in Tab. 3 and Tab. 4 respectively.

Tab. 3: Stoichiometry

		1	2	3	4	5	6	7	8	9	10
	Process	T	S_I	S_S	S_{NO}	S_{NH}	S_P	X_H	X_A	X_S	X_I
1	Aerobic het. growth			$-1/Y_H$		$-i_x$	$-i_P$	1			
2	Anoxic het. growth			$-1/Y_H$	$-(1-Y_H)/2.86/Y_H$	$-i_x$	$-i_P$	1			
3	Aerobic aut. growth				$1/Y_A$	$-1/Y_A - i_x$	$-i_P$		1		
4	Decay het.							-1		1-fp	fp
5	Decay aut.								-1	1-fp	fp
6	Hydrolysis			1		i_x	i_P			-1	
7	P. precipitation						-1				

Tab. 4: Process rates

	Process	Process rates
1	Aerobic het. growth	$\mu_H * S_S/(K_S+S_S) * S_O/(K_{OH}+S_O)* S_{NH}/(0.01+S_{NH}) * S_P/(0.01+S_P)* X_H$
2	Anoxic het. growth	$\mu_H * S_S/(K_S+S_S)* K_{OH}/(K_{OH}+S_O) * S_{NO}/(K_{NO}+S_{NO})$ $* S_{NH}/(0.01+S_{NH}) * S_P/(0.01+S_P)*\eta g*X_H$
3	Aerobic aut. growth	$\mu_A * S_{NH}/(K_{NH}+S_{NH}) * S_O/(K_{OA}+S_O) * S_P/(0.01+S_P) * X_A$
4	Decay het.	$b_H * X_H$
5	Decay aut.	$b_A * X_A$
6	Hydrolysis	$kh * X_S/X_H/(K_X+X_S/X_H) * X_H$
7	P. precipitation	k_P*S_P

Main differences to the original ASM1 are:

Switching functions of Monod type growth processes: All growth processes (of Monod type) contain switching functions for nitrogen and phosphorus. That is, in the absence of those the growth processes are stopped. This is in accordance with ASM3. The *K* values of the Monod type switching functions are set to 0.01 directly in the software code and not subject to variation.

Ammonification process : The ammonification process is neglected here and all soluble organic ammonia (nitrogen fraction in S_S) is assumed to be directly converted to S_{NH}. Due to the coupling of nitrogen fractions to the organic matter the hydrolysis of particulate organic nitrogen (X_{ND} in ASM1) is modelled indirectly in the hydrolysis process of X_S.

Hydrolysis : The hydrolysis process rate was taken as is from ASM3 (Henze *et al.*, 2000).

Phosphorus precipitation: A very simple phosphorus precipitation process was included. The rate expression is a 1.order removal function without any effect by other components. This should reflect in a crude manner the precipitation process. Biological *P* removal is not included.

Secondary clarifier

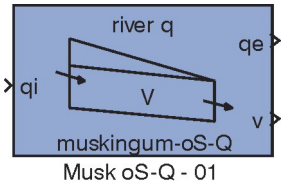
Covering the relevant physical processes in the secondary clarifier under unsteady flow conditions is an essential part of a WWTP-model when used in integrated urban drainage simulations. A one-dimensional settler model is implemented based on solid flux theory. The model background is described in (Takacs et al., 1991) and with respect to details the reader is referred to this publication.

Waste sludge removal

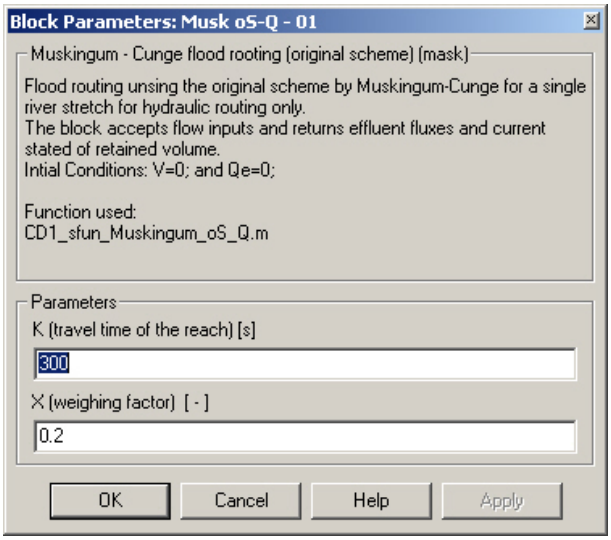
Sewage treatment plants convert a substantial part of the influent waste into bacterial biomass that has to be removed subsequently in order to keep an appropriate concentration level in the reactors. The implementation of waste sludge removal is based on the concept of imposing a certain sludge age as described in ASM 1 (Henze *et al.*, 1987).

7 RIVER (FLOOD ROUTING) BLOCKS

7.1 Muskingum oS - Q



Musk oS-Q - 01



Function:

Muskingum oS-Q describes the flood routing by the Muskingum Method (Roberson *et al.*, 1995a) for a single stretch. **The block considers hydraulics only**, where contaminant transport is not included.

Input:

q_I Upstream inflow in the stretch in [m³/s].

Output:

q_E Downstream outflow from the stretch in [m³/s].
v Current Volume stored in the reach [m³].

Parameters:

K Muskingum parameter describing the time required for a discharge wave travelling through the reach [s].
X Dimensionless weighting factor that relates to the amount of wedge storage [-]. Usually in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).

S-function parameters: [Δt, K, X, C_0,C_1,C_2]

C_0,C_1,C_2 Muskingum constants calculated within the block mask (no user input required). See theory section below.
Δt Sampling rate Δt [s] is inherited from global setting of simulation parameters.

Initial Conditions:

$Q_E = 0$; V = 0; $Q_I = 0$;

Theory:

The outflow from a reach is calculated as a function of $Q_{E,I}$, $Q_{I,i}$ *and* $Q_{E,I+1}$. Index *i* denotes the time step.

$$Q_{E,i+1} = C_0 \cdot Q_{I,i+1} + C_1 \cdot Q_{I,i} + C_2 \cdot Q_{E,i}$$

Muskingum constants C_0,C_1 and C_2 are constant over time and calculated as followed:

$$C_0 = \frac{0{,}5 \cdot \Delta t - K \cdot X}{K \cdot (1 - X) + 0{,}5 \cdot \Delta t} \; ; \quad C_1 = \frac{0{,}5 \cdot \Delta t + K \cdot X}{K \cdot (1 - X) + 0{,}5 \cdot \Delta t} \; ; \quad C_2 = \frac{K \cdot (1 - X) - 0{,}5 \cdot \Delta t}{K \cdot (1 - X) + 0{,}5 \cdot \Delta t}$$

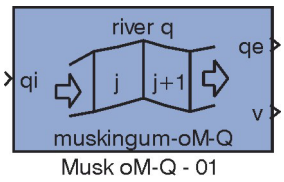
The actual volume stored in the river reach denotes as:

$$V_{i+1} = K \cdot \left[X \cdot Q_{I,i+1} + (1 - X) \cdot Q_{E,i+1}) \right]$$

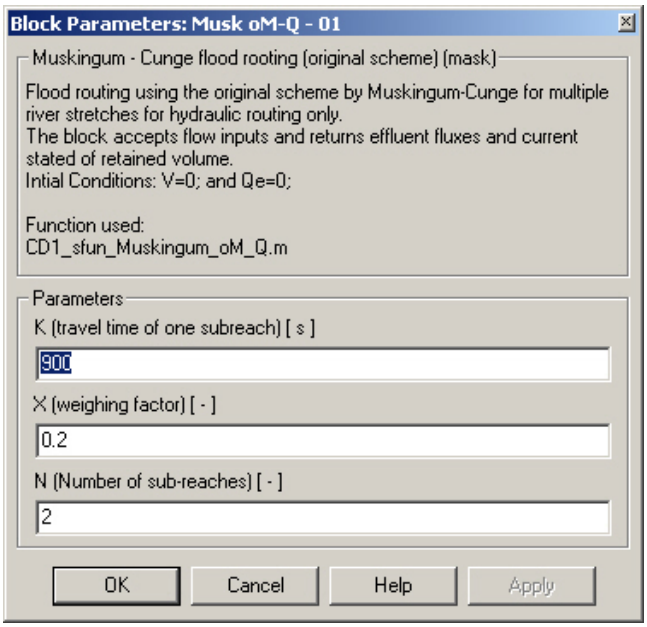
The numerical stability of the Muskingum equation is given when fulfilling the following relation of K, X and ∆t:

$$\frac{1}{2 \cdot (1 - X)} \leq \frac{K}{\Delta t} \leq \frac{1}{2 \cdot X}$$

7.2 Muskingum oM - Q



Musk oM-Q - 01



Function:

Muskingum oM-Q describes the flood routing by the Muskingum Method (Roberson *et al.*, 1995a) for a river stretch comprised holding multiple subreaches. **The block considers hydraulics only**, where contaminant transport is not included.

Input:

q_I Upstream inflow in the stretch in [m³/s].

Output:

q_E Downstream outflow from the stretch in [m³/s]. Outflow from the last (N^{th}) sub reach.

v Vector of current volumes stored in sub reaches [m³]. The length of the Vector depends on the number of sub reaches (N) considered.
[V1, V2,… V_{N-1}, V_N]

Parameters:

K Muskingum parameter describing the time required for a discharge wave travelling through the reach [s]. K applies to one subreach and does not cover travelling time for all sub reaches.

X Dimensionless weighting factor that relates to the amount of wedge storage [-]. Usually in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).

N Number of subreaches

S-function parameters: [Δt, N, K, X, C_0,C_1,C_2]

C_0,C_1,C_2 Muskingum constants calculated within the block mask (no user input required). Constants apply to one subreach !! See theory section below.

Δt *Sampling rate Δt [s] is inherited from global setting of simulation parameters.*

Initial Conditions:

Volumes in and flows rate between subreaches are considered to be zero for the initial conditions.

Theory:

Flows from a sub reaches (j) are driven by the outflow from their upstream sub reach (j-1) where *j* denotes the reach number and *i* denotes the time step.

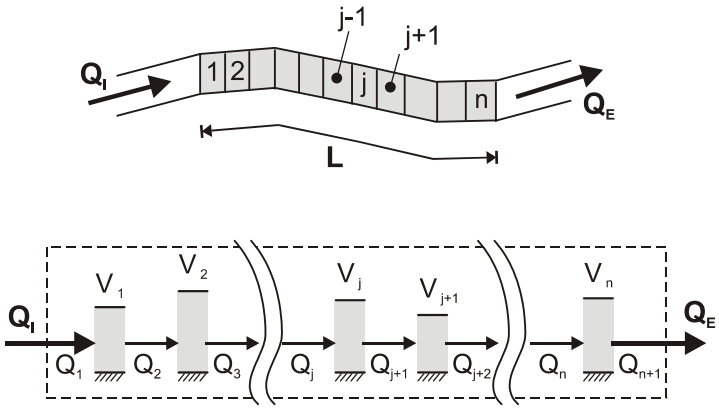


Fig. 20. Schematic on nomenclature for multiple subreaches

$$Q_{i+1}^{j+1} = C_0 \cdot Q_{i+1}^{j} + C_1 \cdot Q_i^{j} + C_2 \cdot Q_i^{j+1}$$

$$C_0 = \frac{0{,}5 \cdot \Delta t - K \cdot X}{K \cdot (1-X) + 0{,}5 \cdot \Delta t}\ ; \quad C_1 = \frac{0{,}5 \cdot \Delta t + K \cdot X}{K \cdot (1-X) + 0{,}5 \cdot \Delta t}\ ; \quad C_2 = \frac{K \cdot (1-X) - 0{,}5 \cdot \Delta t}{K \cdot (1-X) + 0{,}5 \cdot \Delta t}$$

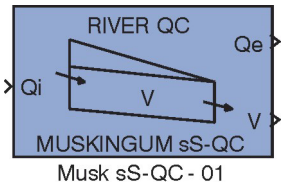
Current Volume stored for a sub reach *j* is calculated as:

$$V_{i+1}^{j} = K \cdot \left[X \cdot Q_{i+1}^{j} + (1-X) \cdot Q_{i+1}^{j+1}) \right]$$

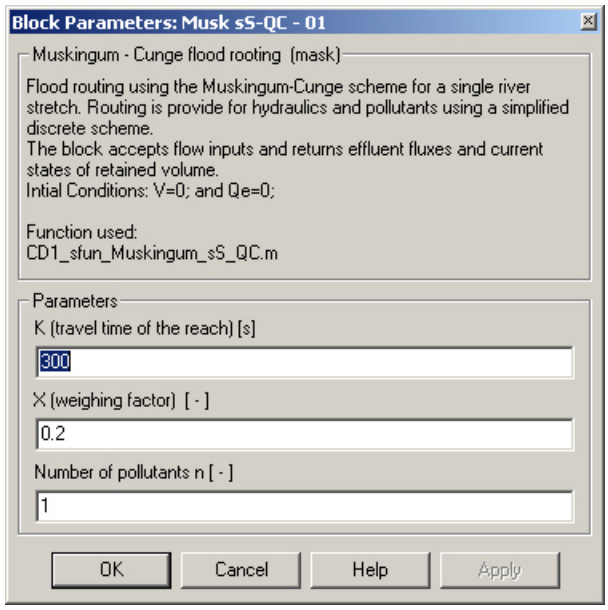
Again, for numerical stability the following relation between K, X and the discrete sampling time Δt must be fulfilled:

$$\frac{1}{2 \cdot (1-X)} \leq \frac{K}{\Delta t} \leq \frac{1}{2 \cdot X}$$

7.3 Muskingum sS - QC



Musk sS-QC - 01



Function:

Muskingum sS-QC describes the flood routing by the Muskingum Method. Formulation is based on a simplified discrete scheme compared to the original scheme. Instead of considering the discharges Q at instant times they are considered as mean discharges over the last discrete period of time. This is feasible when recalling that the measured precipitation represents cumulated (mean) quantities of rainfall for discrete time periods.
The block provides hydraulic routing as well as routing of pollutants.

Input:

Q_I Upstream inflow in the stretch in [m³/s].

Output:

Q_E Downstream outflow from the stretch. [q_E, $C_{E,1}$, $C_{E,2}$, ….,$C_{E,np}$].
V Vector of current volumes and pollutant concentrations stored in the reach. [V, C_1, C_2,…, C_{np}]

Parameters:

K Muskingum parameter describing the time required for a discharge wave travelling through the reach [s].
X Dimensionless weighting factor that relates to the amount of wedge storage [-]. Usually in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).
n_P Number of pollutants carried

S-function parameters: [Δt, K, X, C_A,C_B,n_P]

C_A,C_B Muskingum constants calculated within the block mask (no user input required). See theory section below.
Δt *Sampling rate Δt [s] is inherited from global setting of simulation parameters.*

Initial Conditions:

Flows and concentrations in the reach are initially considered being zero.

Theory:

The outflow from a reach is calculated as a function of $Q_{I,i}$ *and* V_{i-1}. Index *i* denotes the corresponding time step, where flow rates *Qi* represent mean values of flow of the past time step $\Delta t_i = t_i - t_{i-1}$.

$$Q_{E,i} = \frac{Q_{I,i} \cdot C_A + V_{i-1}}{C_B}$$

with

$$C_A = \frac{\Delta t}{2} - K \cdot X$$

$$C_B = \frac{\Delta t}{2} + K \cdot (1 - X)$$

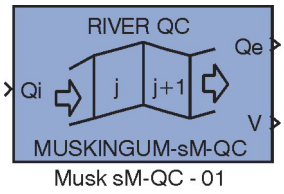
The actual volume stored in the river reach denotes as:

$$V_i = \left(Q_{I,i} - Q_{E,i}\right) \cdot \Delta t + V_{i-1}$$

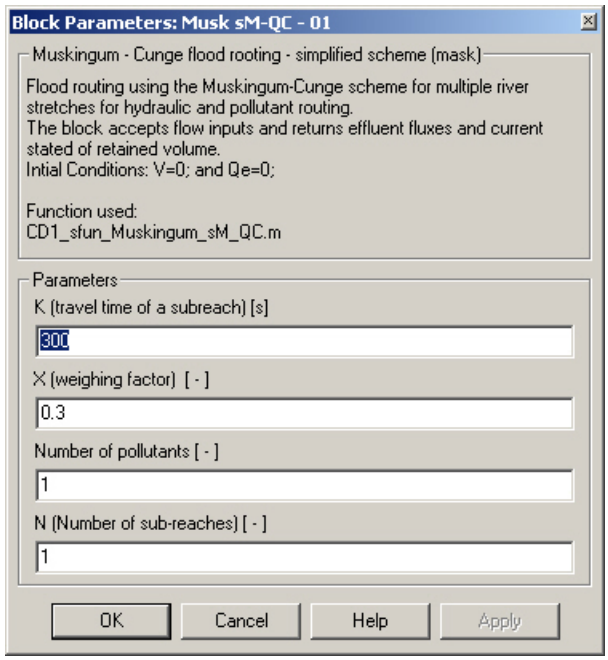
Again, for numerical stability criteria defining the relation between K, X and the discrete sampling time Δt must be fulfilled:

$$1 \le \frac{K}{\Delta t} \le \frac{1}{2X}$$

7.4 Muskingum sM - QC



Musk sM-QC - 01



Function:

Muskingum sM-QC describes the flood routing by the Muskingum Method. Formulation is based on a simplified discrete scheme equivalent to the formulation used in Muskingum sS-QC. Instead of considering the discharges Q at instant times they are considered as mean discharges over the last discrete period of time.

The block provides hydraulic routing as well as routing of pollutants using a user defined number of multiple sub reaches.

Input:

Q_I Upstream inflow in the river stretch being the flow into the first sub reach.
$[q_I, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$

Output:

Q_E Downstream outflow from the river stretch $[q_E, C_{E,1}, C_{E,2}, \ldots, C_{E,np}]$.

V Vector of current volumes and pollutant concentrations stored in each of the N sub reaches of the river stretch.
$[V^1, C_1^1, C_2^1, \ldots, C_{np}^1, \ldots, V^j, C_1^j, C_2^j, \ldots, C_{np}^j, \ldots, V^N, C_1^N, C_2^N, \ldots, C_{np}^N]$

Parameters:

N Number of subreaches

K Muskingum parameter describing the time required for a discharge wave travelling through the reach [s]. **K applies to one subreach** and does not cover travelling time for all sub reaches.

X Dimensionless weighting factor that relates to the amount of wedge storage [-] in the range of 0 (linear reservoir storage) and 0.5. (Typical value = 0,2).

n_P Number of pollutants carried

S-function parameters: $[\Delta t, K, X, C_A, C_B, n_P, N]$

C_A, C_B Muskingum constants calculated within the block mask (no user input required). See theory section below.

Δt Sampling rate Δt [s] is inherited from global setting of simulation parameters.

Initial Conditions:

Flows and concentrations in the reach are initially considered being zero.

Theory:

The outflow from a sub reach Q^{j+1} is calculated as a function of the inflow Q^j_i *and stored volume* V^j_{i-1}. Index *i* denotes the corresponding time step, where flow rates *Qi* represent mean values of flow of the past time step $\Delta t_i = t_i - t_{i-1}$. Index j denotes the number of the sub reach.

$$Q_i^{j+1} = \frac{Q_i^j \cdot C_A + V_{i-1}^j}{C_B}$$

with

$$C_A = \frac{\Delta t}{2} - K \cdot X$$

$$C_B = \frac{\Delta t}{2} + K \cdot (1 - X)$$

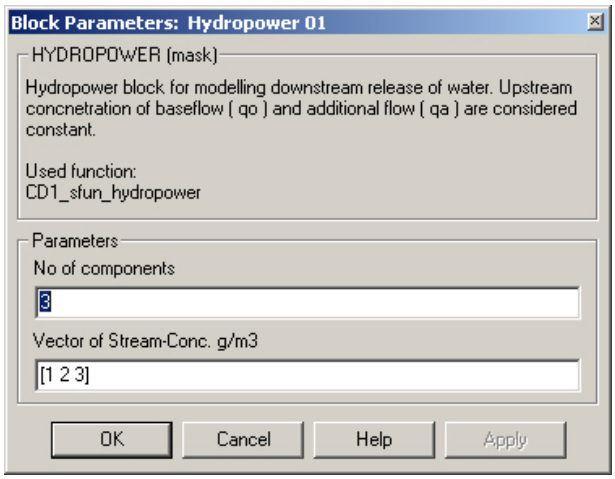
The actual volume stored in a sub reach denotes as

$$V_i^j = \left(Q_i^j - Q_i^{j+1}\right) \cdot \Delta t + V_{i-1}^j$$

Again, for numerical stability criteria defining the relation between K, X and the discrete sampling time Δt must be fulfilled:

$$1 \le \frac{K}{\Delta t} \le \frac{1}{2X}$$

7.5 Hydropower



Function:

The block simulates river discharges from retaining structure such as hydropower intakes. The flow discharged downstream comprises of flow rates for baseflow q_o and additional discharge q_a. Background concentrations associated are introduced as constants in the block mask.

Input:

q_o	Flowrate of baseflow [m³/s]
q_a	Flowrate of additional flow [m³/s]

Output:

Q_E	Outflow from the Hydropower block [q_E, $C_{E,1}$, $C_{E,2}$, ….,$C_{E,np}$].

Parameters:

n_P	Number of pollutants carried
C_{STREAM}	Constant pollutant background concetration [C_1, C_2, ….,C_{np}].

S-function parameters: [n_P, C_{STREAM} , Δt]

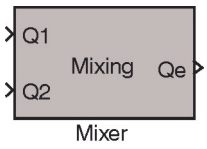
Δt	Sampling rate Δt [s] is inherited from global setting of simulation parameters.

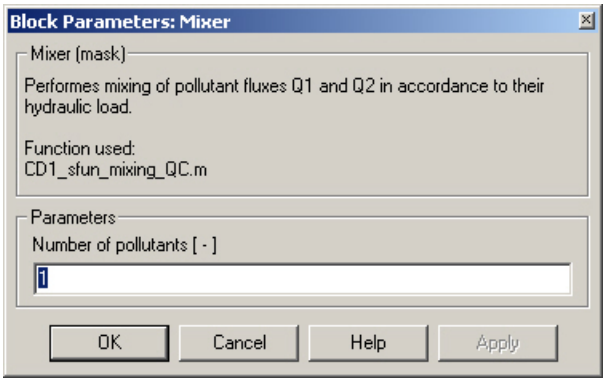
Initial Conditions:

No initial conditions required.

8 TOOLS

8.1 Mixing QC





Function:

The block is designed to combine flows from two sources by means of simple mixing.

Input:

Q_1 Dynamic input of flow and pollutant concentrations associated $[q_1, C_{1,1}, C_{1,2}, \ldots, C_{1,np}]$

Q_2 Dynamic input of flow and pollutant concentrations associated $[q_2, C_{2,1}, C_{2,2}, \ldots, C_{2,np}]$

Output:

Q_E Outflow from the block $[q_E, C_{E,1}, C_{E,2}, \ldots, C_{E,np}]$.

Parameters:

n_P Number of pollutants carried

S-function parameters: $[n_P, \Delta t]$

Δt *Sampling rate Δt [s] is inherited from global setting of simulation parameters.*

Initial Conditions:

No initial conditions required.

Theory:

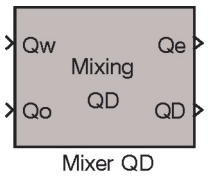
Equation for mixing of substances

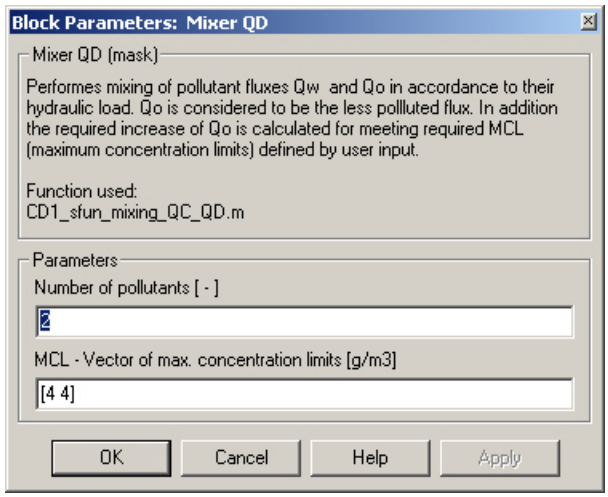
$$C_E = \frac{C_1 \cdot q_1 + C_2 \cdot q_2}{q_E}$$

with

$$q_E = q_1 + q_2$$

8.2 Mixing QC-QD





Function:

The block is designed to combine flows from two sources by means of simple mixing. Two inflows Q_0 and Q_W are considered where Q_0 is considered being the less polluted flow. In extend to the block Mixing-QC the block evaluates the additional flow rates q_D required to meet user defined maximum concentration limits C_{MCL}.

A standard application of the block is to combine upstream base flow in a river (Q_0) and combined sewer overflow (Q_W) entering the river.

Block Type:

Discrete Block utilizing S-function: *CD1_sfun_mixing_QC_QD.m*

Input:

Q_w Dynamic input of flow and pollutant concentrations associated $[q_W, C_{W,1}, C_{W,2}, \ldots, C_{W,np}]$

Q_0 Dynamic input of flow and pollutant concentrations associated $[q_0, C_{0,1}, C_{0,2}, \ldots, C_{0,np}]$

Output:

Q_E Outflow from the block $[q_E, C_{E,1}, C_{E,2}, \ldots, C_{E,np}]$.

Q_D Vector of flow rate demands for each pollutant concentration respectively. $[q_{D1}, q_{D2}, q_{D3}, q_{D4}, \ldots q_{D,np}, q_{D,MAX}]$. The maximum flow rate demand out of all is stored in the vectors last entry.

Parameters:

n_P Number of pollutants carried

C_{MCL} Maximum concentrations limits defined by the user $[C_{MCL,1}, C_{MCL,2}, \ldots, C_{MCL,np}]$

S-function parameters: $[n_P, \Delta t, C_{MCL}]$

Δt *Sampling rate Δt [s] is inherited from global setting of simulation parameters.*

Initial Conditions:

No initial conditions required.

Theory:

Mixing of substances is done equivalent as in the block *Mixing-QC*.

$$C_E = \frac{C_1 \cdot q_1 + C_2 \cdot q_2}{q_E}$$

with

$$q_E = q_1 + q_2$$

Outflow concentrations are checked for compliance with user defined maximum concentration limits (C_{MCL}). In case of non-compliance the additional upstream flow (q_0) for sufficient dilution is returned. Based on the extended mixing formula

$$C'_E = \frac{q_W \cdot C_W + (q_0 + q_D) \cdot C_0}{q_W + q_0 + q_D} \leq C_{MCL}$$

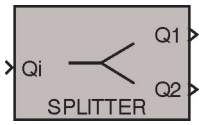
the additional upstream demand of flow rate denotes as

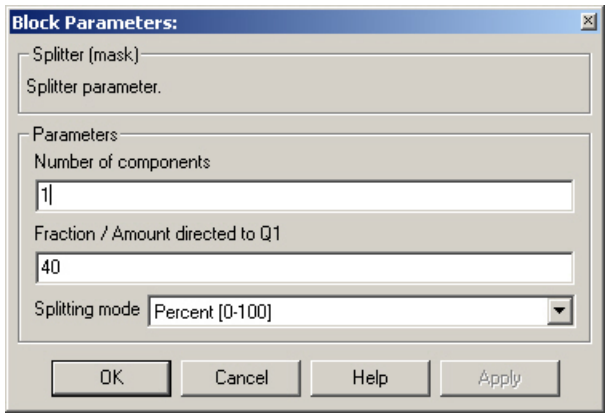
$$q_D \geq \frac{q_W \cdot (C_{MCL} - C_W) + q_0 \cdot (C_{MCL} - C_0)}{(C_0 - C_{MCL})}$$

For having different substances considered, the minimum demanding flow rate is returned for each substance as well as the maximum out of these:

$Q_D = [q_{D1}, q_{D2}, q_{D3}, q_{D4}, \ldots q_{D,np}, q_{D,MAX}]$. with

8.3 Splitter





Function:

The block is designed to split flows according to a fixed flows rate of factor. Thus, the block may be used to model a pump operating with a fixed flow rate using the splitting mode *{ m³/s }* Alternative structures that divert the inflow into two flows at a specific ration may be modelled using either the splitting mode *{ Percent (0-100) }* or *{ Fraction (0-1) }*.

Input:

Q_I Dynamic input of flow and pollutant concentrations associated [q_I, $C_{I,1}$, $C_{I,2}$, ….,$C_{I,np}$]

Output:

Q_1 Outflow from the block; Fraction according to user input defined in the mask [q_1, $C_{I,1}$, $C_{I,2}$, ….,$C_{I,np}$].

Q_2 Outflow from the block; Remaining fraction $q_I = q_I - q_1$ [q_2, $C_{I,1}$, $C_{I,2}$, ….,$C_{I,np}$].

Parameters:

n_P Number of pollutants carried

f Fraction / amounted directed to Q_1

mode Splitting mode to be used

{ m³/s }	Fixed flow rate diverted to output port Q_1
{ Percent (0-100) }	Fixed percentage of flow diverted to output port Q_1
{ Fraction (0-1) }	Fixed fraction of flow diverted to output port Q_1

S-function parameters: [n_P, f, mode, Δt]

Δt Sampling rate Δt [s] is inherited from global setting of simulation parameters.

Theory:

Splitting of flows is done according to the splitting mode selected where the flow q_1 is calculated as:

Mode { m³/s }

$f \le q_I \rightarrow q_1 = f$ and $q_2 = q_I - q_1$

$f \ge q_I \rightarrow q_1 = q_I$ and $q_2 = 0$

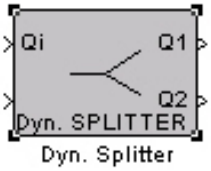
Mode { Percent (0-100) }

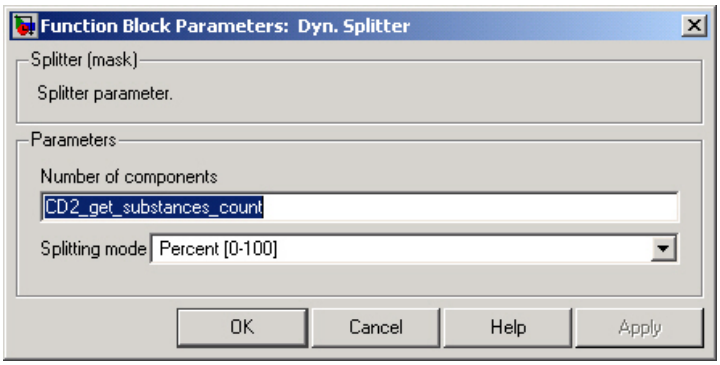
$$q_1 = \frac{f}{100} \cdot q_I \text{ and } q_2 = \left(1 - \frac{f}{100}\right) \cdot q_I$$

Mode { Fraction (0-1) }

$$q_1 = f \cdot q_I \text{ and } q_2 = (1-f) \cdot q_I$$

8.4 Dynamic Splitter



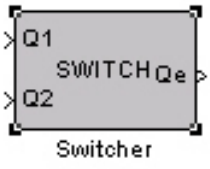


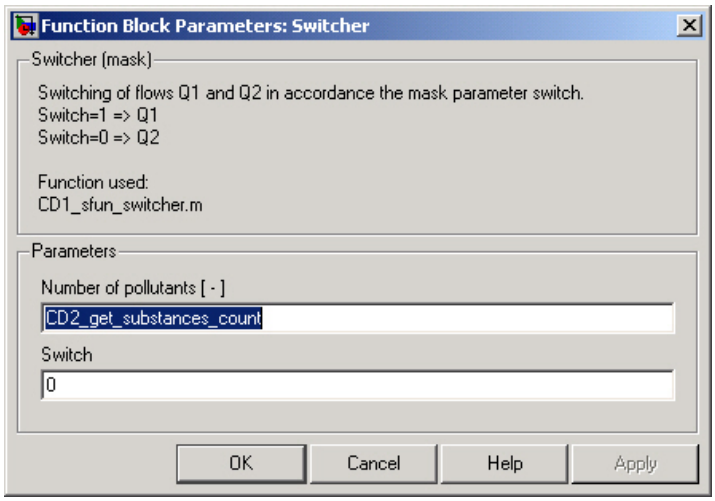
Function:

The block is in its ability to split fluxes equivalent to the original Splitter Block. As the numerical value of the fraction diverted to Q1 is introduced as dynamic input to the “DYN” port, the magnitude of the fraction can be modified during the simulation run. Thus, the block is utilizable for real time control actions.

See Splitter Block for details on the splitting modes and units used.

8.5 Switch





Function:

The block is designed to selective forward either Q1 or Q2 to the output port.

Input:

Q_1 Dynamic input of flow and pollutant concentrations associated $[q_I, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$

Q_2 Dynamic input of flow and pollutant concentrations associated $[q_I, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$

Output:

Q_e Outflow from the block;
Switch=0 → $Q_e = Q_1$
Switch=1 → $Q_e = Q_2$

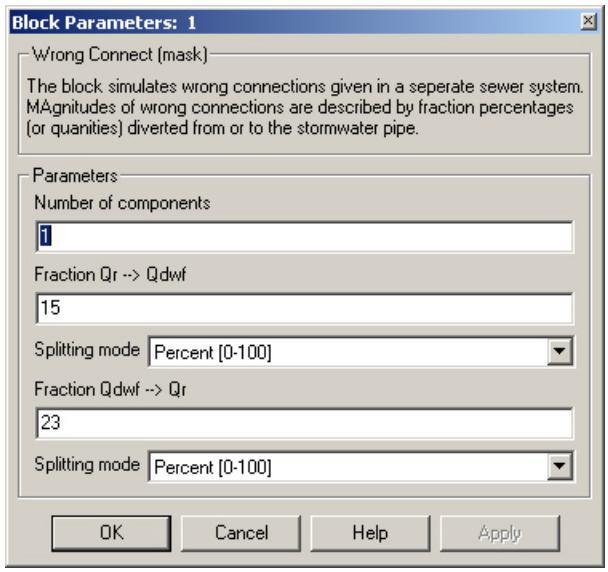
Parameters:

Switch 0/1 defined wether Q_1 of Q_2 is forwarded.

Initial Conditions:

No initial conditions required.

8.6 Wrong Connect



Function:

The block is designed to model wrong connections that may occur in a separate sewer system. Thereby single households may be wrongly connected to the storm water sewer. On the other hand, storm water inlets may be connected to the waster water sewer.
Modelling of such wrong connections is done on a catchment level. It is not the number of wrong connections being of interest, but the overall fraction or quantity of flow in the catchment that is diverted from the storm to the waste sewer (and vice versa).

Input:

Q_r Dynamic input of flow and pollutant concentrations representing storm water flow in the catchment. [q_r, $C_{r,1}$, $C_{r,2}$, ….,$C_{r,np}$]

Q_{dwf} Dynamic input of flow and pollutant concentrations representing dry weather flow in the catchment. [q_{dwf}, $C_{dwf,1}$, $C_{dwf,2}$, ….,$C_{dwf,np}$]

Output:

Q_r Flow and pollutant concentrations in the storm water pipe including wrong connections. [q_r, $C_{r,1}$, $C_{r,2}$, ….,$C_{r,np}$]

Q_{dwf} Flow and pollutant concentrations in the waste water pipe including wrong connections. [q_{dwf}, $C_{dwf,1}$, $C_{dwf,2}$, ….,$C_{dwf,np}$]

Parameters:

n_P Number of pollutants carried

$f_{r\text{-}dwf}$ Fraction / Amounted directed from the storm to the waste water pipe

$m_{r\text{-}dwf}$ Splitting mode to be used (see block Splitter for details)

$f_{dwf\text{-}r}$ Fraction / Amounted directed from the waste to the storm water pipe

$m_{dwf\text{-}r}$ Splitting mode to be used (see block Splitter for details)

Initial Conditions:

No initial conditions required.

Theory:

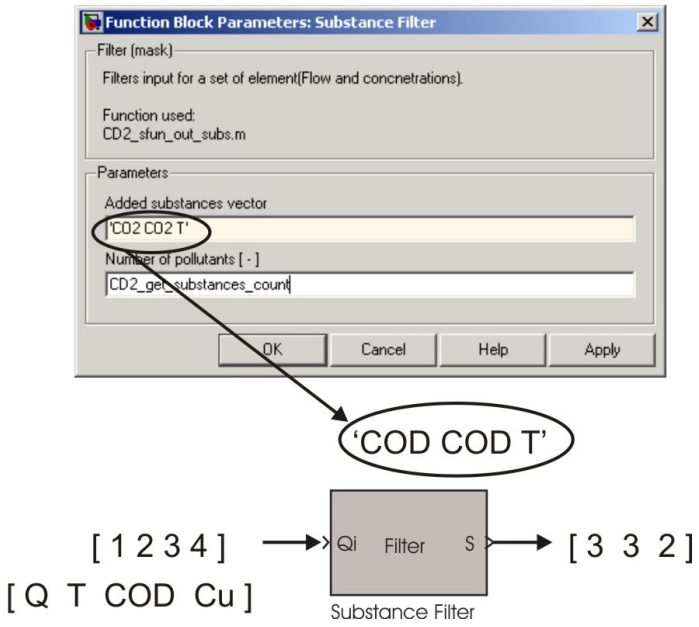
See blocks

Splitter: *CD1_sfun_splitter.m*

Mixer-QC: *CD1_sfun_mixing_QC.m*

for details.

8.7 SetSubstance Q/C



Function:

The block is designed to filter a flow Qi for one or more substances. As the dynamic input Qi to the block does not contain a substance specification, the respective names of substances to filter are to be given as mask input (subs).

In the above example, the flow is filtered for substances 'COD COD T'.

Input:

Q_I Dynamic input of flow and pollutant concentrations associated $[q_I, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$

Output:

S Outflow from the block;

Parameters:

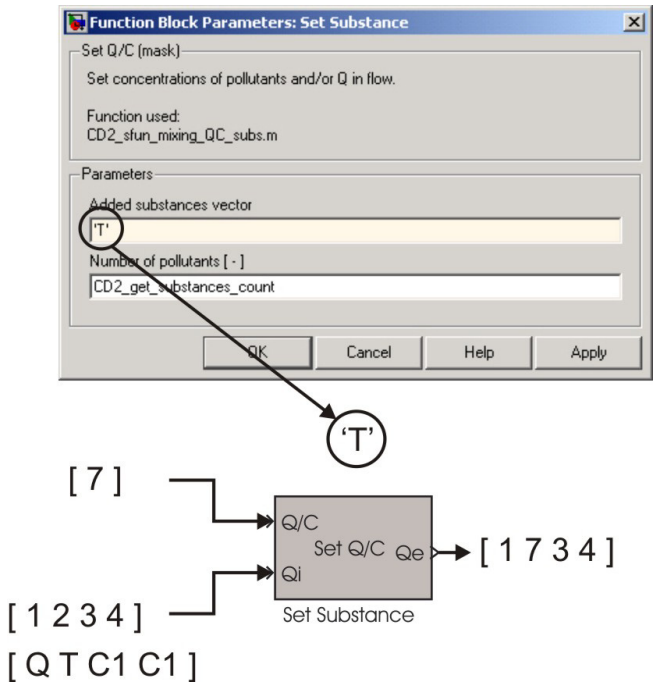
subs String specifying the substances to be filtered

n_P Number of pollutants carried

Initial Conditions:

No initial conditions required.

8.8 Filter



Function:

The block is designed to accept a flow Qi and set one or more substances to a specific value. The vector of values is given as dynamic input Q/C. As the dynamic input to the block does not contain a substance specification, the respective names of substances to be substituted are to be given as mask input (subs).

In the above example, the temperature T is substituted from 2 → 7.

Input:

Q_I Dynamic input of flow and pollutant concentrations associated
$[q_I, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$

Output:

Q_e Outflow from the block; Fraction according to user input defined in the mask
$[q_1, C_{I,1}, C_{I,2}, \ldots, C_{I,np}]$.

Parameters:

subs String specifying the substances to be substituted
n_P Number of pollutants carried

Initial Conditions:

No initial conditions required.

9 LITERATURE

Achleitner S., DeToffol S., Engelhard C. and Rauch W. (2005). The European Water Framework Directive: Water Quality Classification and Implications to Engineering Planning. *Environmental Management*, **35** (4),517-525.

Achleitner S. (2006). *Modular conceptual modelling in urban drainage - development and application of CITY DRAIN.* PhD thesis, Unit of Environmental Engineering, Institute of Infrastructure, University of Innsbruck, Faculty of Civil Engineering Science.

Achleitner S., Möderl M. and Rauch W. (2006). CITY DRAIN © - an open source approach for simulation of integrated urban drainage systems. *Environmental Modelling & Software*, (accepted)

Blöch H. (1999). The European Water Framework Directive: Taking European water policy into the next millennium. *Water Science and Technology*, **40** (10),67-71.

Harremoës P. and Rauch W. (1996). Integrated design and analysis of drainage systems, including sewers, treatment plant and receiving waters. *Journal of Hydraulic Research*, **34** (6)

Henze M., Grady Jr. C. P. L., Gujer W., Marais G. v. R. and Matsuo T. (1987). *Activated Sludge Model No. 1. - Scientific and Technical Report.* IAWQPRC, London.

Henze M., Gujer W., Mino T. and van Loosdrecht M. (2000). *Activated Sludge Models ASM1, ASM2, ASM2d and ASM3. IWA Scientific and Technical Report No. 9.* IAWQPRC, London, UK.

Kalman R. E. (1960). A new approach to linear filtering and prediction problem. *Journal of Basic Engineering*, **81D**,35-45.

Lau J., Butler D. and Schütze M. (2002). Is combined sewer overflow spill frequency/volume a good indicator of receiving water quality impact? *Urban Water*, **4**,181–189.

Lijklema L. (1995). Water Quality Standards: Sense and Nonsense. *Water Science and Technology*, **31** (8),321-327.

Motiee H., Chocat B. and Blanpain O. (1997). A storage model for the simulation of the hydraulic behaviour of drainage networks. *Water Science and Technology*, **36** (8-9),57-63.

Rauch W. and Harremoës P. (1996). The importance of the treatment plant performance during rain to acute water pollution. *Water Science and Technology*, **34** (3-4),1–8.

Rauch W. (1997). *RUMBA - waste water treamtment plant modelling (User Manual)*. Institute of Environmental Science and Engineering, Technical University of Denmark, Lyngby Denmark.

Rauch W. and Harremoës P. (1997). Acute pollution of recipients in urban areas. *Water Science and Technology*, **36** (8-9),179–184.

Rauch W., Bertrand-Krajewski J. L., Krebs P., Mark O., Schilling W., Schütze M. and Vanrolleghem P. A. (2002). Deterministic modelling of integrated urban drainage systems. *Water Science and Technology*, **45** (3),81-94.

Roberson J. A., Cassidy J. and Chaudhry M. H. (1995a). *Hydraulic Engineering*. John Wiley Sons, Inc., New York.

Roberson J. A., Cassidy J. J. and Chaudhry M. H. (1995b). *Hydraulic Engineering*. John Wiley Sons, Inc., New York.

Schilling W. and Hartwig P. (1988). Simulation von Reinigungsprozessen in Belebungsanlagen mit Mischwasserzufluß. *gwf wasser-abwasser*, **129** (8)

Schreider S. Y., Young P. C. and Jakeman A. J. (2001). An Application of the Kalman Filtering Technique for Streamflow Forecasting in the Upper Murray Basin. *Mathematical and Computer Modelling*, **33**,733-743.

Takacs I., Patry G. G. and Nolasco D. (1991). A dynamic model of the clarification-thickening process. *Water Research*, **25** (10),1263-1271.

The Mathworks (2003). *Simulink - Writing S-Functions*. Model-Based and System-Based Design, The Mathworks,