EPANET Backtracking Extension (BTX)
User's Manual (Version 1.0)

Feng Shang, James G. Uber
Department of Civil and Environmental Engineering
University of Cincinnati, P.O. Box 210071, Cincinnati, Ohio 45221-0071

August 22, 2009

Introduction

Various computer models have been developed to simulate the change of disinfectant species concentration within water distribution systems. These models can be divided into two general categories [1]: Eulerian and Lagrangian. Eulerian models divide the network into a series of fixed control elements and record the changes at the boundaries and within these elements, while Lagrangian models track changes of discrete parcels of water as they travel through the network. Traditional models are efficient in modeling water quality at all nodes over time, but forget the internal details of the links between water qualities at upstream and downstream locations (or, at least, they are not available in a useful form). These internal details - the input-output (I/O) information - include the number of flow paths, their time delays, and their impact on water quality. Such I/O information is critical to certain applications. One such example is the design of feedback control algorithms used to regulate disinfectant injections based on measurements at a distributed set of sensor locations [2]. Such an application assumes that disinfectant injection can be related to output concentration, using either an I/O or state space model. For water distribution systems, the state space dimension can preclude real-time use of a state space model for computational reasons. Input-output information is also important to determine the location and time of a contaminant intrusion, accidental or intentional, into the water distribution systems. When water quality contamination is detected at a particular location, fault diagnosis should focus on areas that are hydraulically connected to the problem location. More generally, I/O information is required to explain water quality transformation processes that are *path-dependent*, such as those that involve pipe material interactions.

Zierolf et al. [3] first developed an I/O model for chlorine transport in networks without storage tanks. Their algorithm tracks the travel of water parcels in networks in reverse time. This model can find all the paths from the input to the output and the corresponding

delays and was used to calibrate pipe chlorine reaction rates off-line. The particle backtrack algorithm (PBA) [4] extends the work of Zierolf et al. to consider the existence of tanks and to allow multiple water sources and quality inputs. Unlike Zierolf et al., who track each parcel individually and sequentially from output to input, the PBA tracks simultaneously a large number of water parcels, moving them simultaneously along their paths (in reverse time). This change results in a simpler algorithm to describe, and one that is likely more efficient because the algorithm moves in parallel with the time variation in the hydraulic solution.

The potential research and engineering applications motivate the extension of EPANET [5] to include PBA so that user can have easy access to the detailed I/O information that otherwise are not convenient to obtain.

Overview of Particle Backtracking Algorithm

Under the assumption of first order reaction, water quality, such as chlorine concentration, at a downstream location (output) can be expressed as a linear combination of water quality strength (concentration, mass input rate, or strength of secondary water quality input such as booster chlorination) at some upstream locations and time (inputs), weighted by water quality impact coefficients that reflect water dilution at junctions and storage tanks and water quality change due to reaction. Mathematically, such linear relationship can be written as:

$$c_o(T_o) = \sum_{j=1}^{N} \gamma_j s_j (T_o - t_j) \tag{1}$$

where N = number of flow paths between water quality inputs and output; j = flow path index; T_o = output time; s_j = water quality input strength for path j; t_j = water travel time or time delay for flow path j; and γ_j = water quality impact coefficient. The impact coefficient γ_j is the sensitivity of output concentration (at time T_o)to input strength (at time $T_o - t_j$), of which the unit depends on the type of water quality input. Value of the impact coefficients represents the decay or growth due to chemical reaction and dilution at the network junctions and storage tanks. It should be noted that the N inputs must account for all the water the output receives, otherwise Equation 1 is not valid. The number of inputs, N, can be larger than the number of water quality input locations because it is possible that there exist multiple flow paths between an input location and an output and these flow paths are associated with different time delays

Particle backtracking algorithm tracks the water reaching the output until all the water parcels are traced back to either true water source or booster water quality source that fixes the concentration of any flow leaving such "set point" source. Backtracking stops at the "set point" booster source because the output concentration is not sensitive to water quality input upstream of the "set point" water quality input.

Sometime it is interesting to know what is the impact of constant water quality input during a certain period of time on output water quality. The aggregated impact of an water quality source during a period of time can be measured with the sum of impact coefficients. If a water quality output at time T_o is hydraulically connected to a source location with constant strength, s, at n different times $T < t_1 < t_2 \cdots < t_n < T + \delta t$, the output water quality can be calculated as:

$$c_o(T_o) = \sum_{j=1}^{n} (\gamma_j s_j(t_j)) = \sum_{j=1}^{n} (\gamma_j) s = \gamma s \tag{2}$$

where γ is the aggregated impact coefficient of the source between T and $T + \delta T$, with the assumption of constant water quality strength during that period. It is the sum of the impact coefficients of multiple flow paths with time delays between T and $T + \delta T$ from the source to the output.

EPANET-BTX is a software tool through which users can have access to the information that can be obtained with particle backtracking algorithm, which includes number of flow paths, time delay and impact coefficient associated with flow paths, and aggregated impact coefficients of a source (during a certain period) on specific water quality output. EPANET-BTX can only track a single species that follows the first order reaction kinetics, i.e.

$$\frac{dc}{dt} = kc \tag{3}$$

where c = the concentration of the component to be tracked and k = first order rate coefficient. The reaction rate coefficient k, which can be zero for non reactive constituent, depends on both the bulk reaction and wall reaction coefficients, which are specified in the EPANET input file. The details of EPANET input file are in [5].

Program Usage

EPANET-BTX is distributed as a compressed archive file named epanetbtx.zip which contains the following files:

- epanetbtx.dll: EPANET-BTX dynamic linkage library that should be used in conjunction with the EPANET Toolkit library (epanet2.dll).
- epanetbtx.lib: Microsoft C/C++ library file for epanetbtx.dll.
- epanetbtx.h: C/C++ header file for EPANET-BTX.
- epanetbtx.pdf: EPANET-BTX users manual.
- epanet2.h: header file for EPANET toolkit.

- epanet2.dll: dynamic linkage library of EPANET toolkit.
- epanet2.lib: Microsoft C/C++ library file for epanet2.dll.

Example Toolkit Usage

EPANET-BTX function library must be used in conjunction with the EPANET Programmer's Toolkit which is slightly modified from the original one. EPANET toolkit functions are called to open EPANET input file, run hydraulic simulation and save hydraulic simulation results for backtracking modeling. Application can be written in any programming language that can call Windows DLL functions. The following is an example that is written in C language and shows how EPANET-BTX toolkit function can be used. The next section lists all the functions included in the BTX toolkit.

The program includes two header files: `epanet2.h` and `epanetbtx.h` that are C/C++ header files for EPANET Programmer's Toolkit and Backtracking Extension Tookit, respectively. The program begins with the EPANET Programmer Toolkit function `ENopen` to open an EPANET input file and read in all the network data. Other EPANET Programmer's Tookit functions can be called after `ENopen` to obtain and change network data, such as simulation duration, hydraulic time step, bulk decay coefficient, etc. In the example program, EPANET Tooklit function `ENsettimeparam` is called to set the simulation length to be 240 hours. `ENsolveH` and `ENsavehydfile` are then called to solve network hydraulics and save the hydraulic solution in a binary file. The function `BTXopen` is used to associate hydraulics data with backtracking modeling by specifying the name of a hydraulic binary file. If a binary hydraulic file is available from some early application, functions `ENsolveH` and `ENsavehydfile` are not needed. `BTXopen` is followed by function `BTXinit`, which allocates the memory for backtracking modeling and reads in hydraulic data from the binary hydraulic file specified with `BTXopen`. The function `BTXinit` has two input arguments: the first is the time step in hours for impact coefficient aggregation and the second is the impact coefficient thresh hold to delete backtracking particle with small enough impact on output water quality. Function `BTXsetinput` can be used to specify the nodes whose impacts on output water quality are of interests to the modeler, while `BTXsetinputstrength` is used to set the source strength of a true water quality source. In the example, a constant source strength of 2.0 mg/L is assumed for the node with ID "sourceid" ("10"). A node must be set to be an "input" before source strength is assigned, but an input is not necessarily a true water quality source. `BTXsim` is the function to be called for backtracking modeling and the two input arguments are output node id and water quality simulation time. After backtracking modeling for node with ID "outputid" ("32") at "sampletime", `BTXout` is called to calculate the output water quality given water quality source strength set by `BTXsetinputstrengh` and `BTXgetimpact` is called to withdraw the vector of impact coefficients quantifying the water quality impact of the node "sourceid" on node "outputid" at "sampletime". The first argument of the `BTXgetimpact` must be a valid node id that has been specified as an input us-

ing `BTXsetinput`. BTXgetimpact writes the impact coefficients of "sourceid" on "outputid" at "sampletime" into the "impactvector" with length "impactvectorlength" : because the impact time step is set to be 1.0 hour, impactvector[n] contains aggregated impact coefficient of constant water quality source at "sourceid" between hour n and n+1 on "outputid" at "sampletime". If impactvector[n] returns value larger than 0, node "sourceid" between hour n and n+1 is hydraulically connected to node "outputid" at "samlpetime", i.e. part of water node "outputud" receives at "samlpetime" pass through node "soueceid" sometime between hour n and n+1. Function `BTXgetpathnumber` obtains the number of flow paths (backtracked), which starts at "outputid" and ends at either water source or SETPOINT secondary water quality source. Also available are toolkit functions to get detailed flow path information, such as time delay and impact coefficient, and to check if water flows through a specific pipe before reaching the output (see next section for details). The functions `BTXsim`, `BTXout`, `BTXgetimpact` and functions to obtain flow path information can be called repeatedly to do backtracking modeling and flow path analysis for any node and any time. At the end of the program, function `BTXclose` and EPANET Toolkit function `ENclose` are called in sequence to free the memories used by backtracking application and EPANET.

```
#include"epanetbtx.h"
#include "epanet2.h"

int main()
{
char *epainpfile="net1.inp", *eparptfile="rpt", *outputid="32", *sourceid = "10";
float sampletime = 200.0;
float impactvector[224];
float sourcestrength = 2.0, sourcepatstep = 1.0, samplestep = 1.0, impactstep = 1.0, thresh
= 0.000001, wqres;
int i, sourcepatlength=1, impactvectorlength = 224, pathnumber;

ENopen(epainpfile, eparptfile, "");
ENsettimeparam(EN_DURATION, 240*3600);
ENsolveH();
ENsavehydfile("hyddata");
BTXopen("hyddata");
BTXinit(impactstep, thresh);
BTXsetinput(sourceid, EN_SETPOINT);
BTXsetinputstrength(sourceid, &sourcestrength, sourcepatlength, sourcepatstep);
for( i = 0; i < 24; i++)
{
sampletime = sampletime + samplestep*i;
BTXsim(outputid, sampletime);
BTXout(&wqres);
```

```
BTXgetimpact(sourceid, impactvector, impactvectorlength);
BTXgetpathnumber(&pathnumber);
}
BTXclose();
ENclose();
return(0);
}
```

Backtracking Toolkit Functions

int BTXopen(char *f1)

1. *Description*
   Opens the Backtracking Toolkit and associates a hydraulic solution file with the backtracking modeling

2. *Arguments*
   f1: name of binary EPANET hydraulic file.

3. *Returns*
   Returns an error code if there are errors, otherwise returns 0.

4. *Notes*
   The binary hydraulic file can be generated in previous application or in the current application with EPANET Programmer's Toolkit functions.

int BTXinit(float impactstep, float thresh)

1. *Description*
   Alllocates memory for data structure and read hydraulic data from EPANET hydraulic file.

2. *Arguments*
   impactstep: time step to integrate impact coefficients, in the unit of hour.
   thresh: threshold for particle impact coefficient to delete backtracking particles with small impact on output water quality.

3. *Returns*
   Returns an error code if there are errors, otherwise returns 0.

int BTXsim(char *outputid, float sampletime)

1. *Description*
   Does backtracking water quality modeling at a node with ID outputid and at sampletime.

2. *Arguments*
   outputid: ID of the output node.
   sampletime: backtracking modeling time (in hours).

3. *Returns*
   Returns an error code if there are errors, such as invalid output ID, otherwise returns 0.

int BTXout(float *result)

1. *Description*
   Calculates water quality modeling at the node and time specified by `BTXsim`.

2. *Arguments*
   result: output water quality value (in mg/L).

3. *Returns*
   Returns 0.

4. *Notes*
   Should be called after BTXsim specifies the output node and sampling time and the result also depends on the source strength set by function `BTXsetinputstrength`.

int BTXsetinput(char * nodeid, int type)

1. *Description*
   Sets water quality source location or the node whose impact on output water quality is of interest.

2. *Arguments*
   nodeid: input node ID.
   type: water quality types that are defined in `EPANET2.h`: EN_CONCEN(0), EN_MASS(2), EN_SETPOINT(3) or EN_FLOWPACED(4).

3. *Returns*
   Returns an error code if there are errors, such as invalid node ID, otherwise returns 0.

4. *Notes:*
   The backtracking process stops at any SETPOINT source. So if the objective is to study the impact of multiple nodes on a specific output, do not set the source type to be SETPOINT.

int BTXsetinputstrength(char * nodeid, float * strength, int patternlength, float patternstep)

1. *Description*
   Set water quality source strength.

2. *Arguments*
   nodeid: node ID of water quality source.
   strength: vector to store water quality strength. The index starts with 0.
   patternlength: length of the vector of strength.
   patternstep: strength pattern step in the unit of hour.

3. *Returns*
   Returns error code if there are errors, such as invalid node ID, otherwise returns 0.

4. *Notes:*
   1) The nodeid must be specified as an input using `BTXsetinput` before water quality source strength can be assigned.
   2) The source strength vector repeats itself if the simulation time is longer than the duration that the pattern covers.

int BTXgetimpact(char * nodeid, float * impact, int size)

1. *Description*
   Gets aggregated impact coefficients of an input node on output water quality

2. *Arguments*
   nodeid: ID of the input node.
   impact: vector to store aggregated impact coefficients.
   size: length of the vector impact.

3. *Returns*
   Returns an error code if there are errors, such as invalid node ID, otherwise returns 0.

4. *Notes*
   The vector impact is filled from index 0 to size-1. The size should be larger than the number of impact coefficient integration steps, which is the function of output time and impact coefficient aggregation time step.

int BTXgetpathnumber(int * npath)

1. *Description*
   Gets number of flow paths from true water sources and/or SETPOINT water quality sources to output node at sample time, which are specified by `BTXsim`.

2. *Arguments*
   npath: integer pointer to store number of flow paths.

3. *Returns*
   Returns 0.

int BTXgetpathinfor(int pathindex, int * endnodeindex, float * timedelay, float * impactcoeff)

1. *Description*
   Gets flow path information.

2. *Arguments*
   pathindex: index of flow path, which should be between 1 and number of flow paths that can be obtained with `BTXgetpathnumber`.
   endnodeindex: integer pointer to store the index of the node where the backtracked flow path ends.
   timedelay: the float pointer to store the time delay of the flow path (in hours).
   impactcoeff: impact coefficient of the flow path.

3. *Returns*
   Returns an error code if there are errors, such as invalid path index, otherwise returns 0.

4. *Notes:*
   1) The smaller the flow path index, the shorter the time delay of the path.
   2) Some flow paths end at water sources that are not water quality sources and the impact coefficients of such paths are therefore 0.0. Water quality sources have to be specified using `BTXsetinput`.

int BTXpipeinpath(char * pipeid, int *flag)

1. *Description*
   Checks if a pipe is in one or more flow paths.

2. *Arguments*
   pipeid: ID of a network pipe.
   flag: integer pointer to store 1 if true or 0 otherwise.

3. *Returns*
   Returns an error code if there are errors, such as invalid pipe index, otherwise returns 0.

int BTXclose()

1. *Description*
   Frees the memory used by BTX.

2. *Returns*
   Returns 0.

BTX ERROR CODES

Error 102: no EPANET input file supplied. A standard EPANET input file was not opened with `ENopen` before BTX was opened with `BTXopen`.

Error 601: insufficient memory available. There is not enough physical memory in the computer to do backtracking analysis.

Error 602: can not open EPANET hydraulic file. The binary hydraulic file specified with `BTXopen` either does not exist or can not be opened.

Error 603: hydraulic file error. The binary hydraulic file specified with `BTXopen` is not a valid hydraulic file corresponding to the EPANET input file opened before `BTXopen`.

Error 604: insufficient simulation time. The beginning of the simulation period is reached before all backtracking particles are traced back to sources and there for water quality modeled depends on initial water qualities, which are assumed to be zero for backtracking analysis.

Error 605: invalid source type. Invalid EPANET water quality source types. The source types defined in EPANET header file are EN_CONCEN(0), EN_MASS(1), EN_SETPOINT(2), and EN_FLOWPACED(3).

Error 606: invalid input. A node is not a water quality input. `BTXsetinputstrength` and `BTXgetimpact` return this error code if the node has not been specified as an input using `BTXsetinput`.

Error 607: invalid node ID.

Error 608: invalid link ID.

Error 609: invalid water quality output time. Either less than 0 or larger than the hydraulic simulation duration.

Error 610: invalid path index. Either less than 1 or larger than total number of flow paths.

References

[1] L.A. Rossman and P.F. Boulos. Numerical methods for modeling water quality in distribution systems: A comparison. *Journal of Water Resources Planning and Management, ASCE*, 122(2):137–146, 1996.

[2] Z. Wang, M. M. Polycarpou, and J. G. Uber. Decentralized model reference adaptive control of water quality for water distribution systems. In *Proceedings of 15th IEEE International Symposium on Intelligent Control*, pages 127–132, Rio, Greece, July 2000.

[3] M. L. Zierolf, M. M. Polycarpou, and J. G. Uber. Development and auto-calibration of an input-output model of chlorine transport in drinking water distribution systems. *IEEE Trans. on Control Systems Technology*, 6(4):543–553, July 1998.

[4] F. Shang, J. Uber, and M. Polycarpou. Particle backtracking algorithm for water distribution system analysis. *J. Environ. Eng.*, 128(5):441–450, May 2002.

[5] L. A. Rossman. Epanet 2 users manual epa/600/r-00/057. 2000.