Simulation Program for River Network (SPRNT) User Guide and Tutorial

Cheng-Wei Yu [*], KyungMin Kim[†], Frank Liu [‡] and Ben R. Hodges [§]

2016.10.23

[*]University of Texas at Austin, Center for Research in Water Resources, yuchanway@utexas.edu
[†]University of Texas at Austin, Center for Research in Water Resources, kkim9545@utexas.edu
[‡]IBM Research Branch at Austin, frankliu@us.ibm.com
[§]University of Texas at Austin, Center for Research in Water Resources, Associate Professor

Introduction

SPRNT is a one-dimensional hydraulic model developed by Dr. Ben R. Hodges from University of Texas at Austin and Dr. Frank Liu from IBM research branch (Liu and Hodges, 2014). Unlike traditional commercial hydraulic models, SPRNT model is the first hydraulic model that use the idea from VLSI (very large scale integration) techniques, which were developed for semiconductor circuit analysis and design. This innovation makes SPRNT become the first cross-discipline hydrodynamic model that enable to do the hydrodynamic simulation over a large-scale area (ex. Centennial/Nation-scale). Since SPRNT is a Linux-based model, it does not have fully-feature user interface like other commercial software does. Users will need to change all the parameters/variables/forcings in the code/terminal window.

This user manual will go through every step of executing SPRNT, including installation, data preparation, data preprocess, input file preparation, input file structure, parameter setting, boundary condition setting, commands for running the model and result file structure. There is also a FAQ at the end of this manual, answers for most of common errors/mistakes can be found there. If you are occurring some problems not listed in FAQ section, please feel free to contact Dr. Frank Liu (`frankliu@us.ibm.com`), Cheng-Wei Yu (`yuchanway@utexas.edu`) or Kyungmin Kim (`kkim9545@utexas.edu`) through email.

1 Prerequisite Files

Please make sure you have all prerequisite files downloaded.

1.1 SPRNT Source Code

For Linux system users:

Linux users can directly download the SPRNT source code from:

`https://github.com/frank-y-liu/SPRNT/archive/master.zip`

Or use git command to download the source code.

```
$ git clone https://github.com/frank-y-liu/SPRNT.git
```

For Mac users:

Mac users can download the .dmg file from the following link:

`SPRNT_dmg_file_location` [Maybe put in Frank's Github?]

For Mac users, gfortran package installation is required before the SPRNT installation, please download the appropriate version of gfortran binary package your from the following locaiton:

`https://gcc.gnu.org/wiki/GFortranBinaries#MacOS`

[Note: Please try to install the gfortran in the default location (which is -L/usr/local/gfortran/lib), otherwise, user will require modifying the library link path during the installation]

1.2 Ready to use preprocessor

The ready-to-use data preprocessor is a program written in C++, users can download it from:
`https://github.com/yuchanway/SPRNT_Preprocessor.git`

or use git command

```
$ git clone https://github.com/yuchanway/SPRNT_Preprocessor.git
```

More details about how to use the preprocessor, can be found in ***Appendix A- How to use the Preprocessor***.

2 Setting up SPRNT in your machine (for Linux users only)

1. **./configure in SPRNT directory**
   Users need to configure the SPRNT program before executing the installation. Move to /SPRNT/ directory and use the command in the following to configure the SPRNT code:

```
$ ./configure
```

2. **Install everything you need indicated as checking /compiler/ .... no**
   The configure process will check all the required libraries/packages, users need to install everything listed in the checking.
   [Note:g77 may not be required though it says no. gfotran may function as g77, so users will just need to make sure gfortran is installed]

3. **Install Zlib (zlib-1.2.8)**
   For convenience, zlib and hdf5 should be installed in the same folder. Go to the directory where zlib is located,

```
$ ./configure -prefix=/home/ed/local
$ make check install
```

4. **Install HDF5 (hdf5-1.8.14)**
   Download the HDF5 source code from:
   `https://www.hdfgroup.org/downloads/index.html`
   Then go to the directory where HDF5 is located and configure the HDF5 source code by using the following commands:

```
$ ./configure --with-zlib=/home/ed/local -prefix=/home/ed/local
$ make check install
```

   Related document is also available for reference:
   `https://www.hdfgroup.org/ftp/HDF5/releases/hdf5-1.8.15-patch1/src/unpacked/release_docs/INSTALL`
   [Note, this step may take more than 10 mins]

5. **Install NetCDF4 (netcdf-4.4.0-rc2)**
   Users are able to download the NetCDF4 (version netcdf-4.4.0-rc2) from the following link:
   `https://github.com/Unidata/netcdf-c/releases/tag/v4.4.0-rc2`
   After the download is finished, use terminal to go to the directory where NetCDF4 is located, and execute the following command to install the netcdf library.

```
$ CPPFLAGS=-I/home/ed/local/include LDFLAGS=-L/home/ed/local/lib
./configure -prefix=/home/ed/local
$ make check install
```

6. **Install UMFPACK**

   In this step, some preprocesses are needed:

   (a) Install *lapack* library by using the following command:

   ```
   sudo apt-get install libblas-dev liblapack-dev
   ```

   (b) Install *numpy* and *openblas*, source codes can be found at this website.
   `https://hunseblog.wordpress.com/2014/09/15/installing-numpy-and-openblas`

   (c) After above steps are finished, users would be able to download and install the *SuiteSparse* package from this website:
   `http://faculty.cse.tamu.edu/davis/suitesparse.html`

   After downloading the UMFPACK package, use terminal to go to the directory where UMFPACK is located and executing the following commands:

   ```
   $ make
   $ sudo make install
   ```

   After unpackaging, copy and paste the folder to both SPRNT/Thirdparty/CMLIB and SPRNT/Thirdparty/UF

7. **Install the SPRNT model**

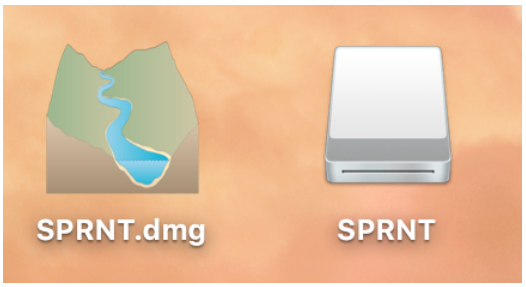
   Go to the SPRNT directory and execute the following commands:

   ```
   $ ./configure
   $ make dep
   $ make
   $ make test
   $ make install
   ```

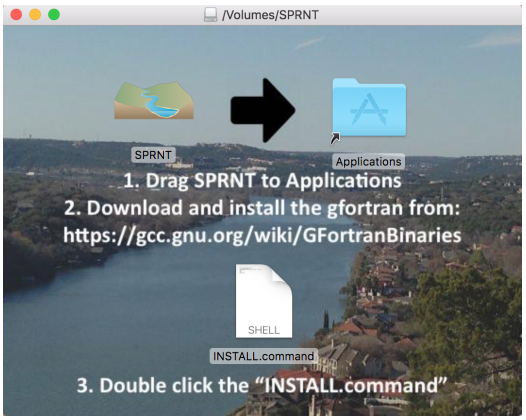
Installation finish.

3 Setting up SPRNT in your machine (for Mac users only)

1. **Mount the dmg file**
   Double clicks on the *SPRNT.dmg* file, a hard drive icon will appear on users' desktop.



2. **Drag the SPRNT into the Applications folder**
   After clicking on the SPRNT hard drive icon, an installation window will pop out. Drag the SPRNT program icon and drop it into the *Applications* folder, like all Mac program installation does.



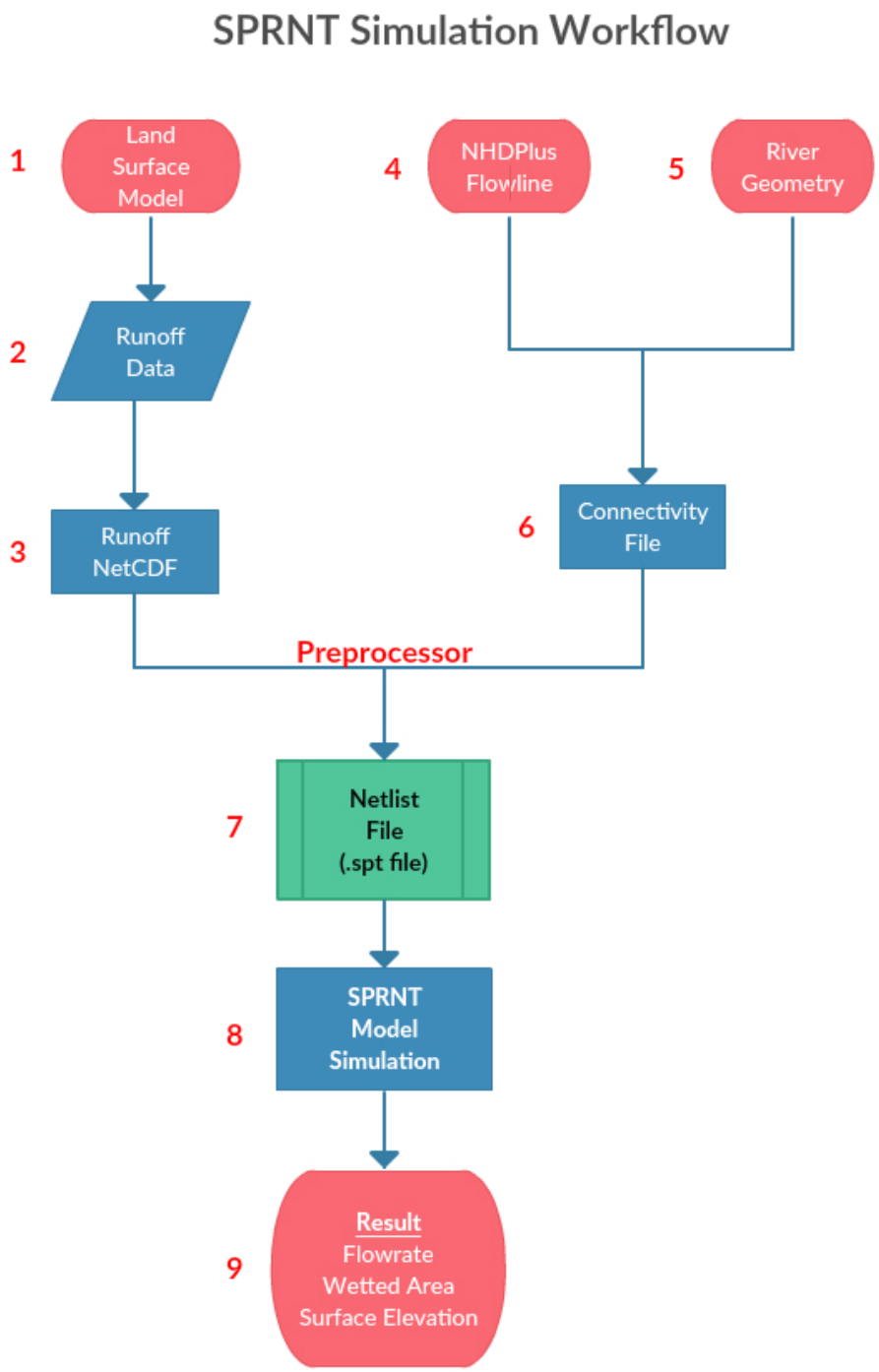
3. **Install the gfortran library**
   Mac users will need to download and install the gfortran package from `https://gcc.gnu.org/wiki/GFortranBinaries`. Install the gfortran binary to the default path (which is under ***/usr/local/***).
   [NOTE]: If users prefer to install the gfortran binary to other place, the link path in the INSTALL.command *file need to be modified.*

4. **Double clicks on *INSTALL.command***
   After the gfortran binary is installed, double clicks on the *INSTALL.command* file. A terminal window will pop out and automatically execute all the required installation commands.

Installation finish.

4 SPRNT Workflow

The workflow chart in here illustrates all the necessary steps for running a successfully SPRNT simulation. Red denotations in the flowchart represent the order of steps. The detail of each step will be elaborated in the following sections.



4.1 Land surface model

In hydraulic/hydrologic modeling, land is an important in component, especially when modeling precipitation and runoff for target river in the study area. Land surface models successfully simulate the land-atmosphere water loop and enable human to predict/approach the precipitation condition in large-scale area. In SPRNT model, we will use the precipitation from land surface models as the lateral flow sources in the model. In here, we made an assumption that the soil in the study area is saturated; so all the effective rainfall from the land surface model will become the river runoff. Land surface models can be: NOAH, NLDAS 2, JULES...etc. However, please aware of that the choice of land surface model will directly influence the accuracy of the hydraulic simulation result.

4.2 Runoff data

In order to successfully run a SPRNT simulation, we need to convert the forcing term to the right format. From the last paragraph, you should get the precipitation data from the land surface mode (normally the unit is mm/hour). Nevertheless, the unit of runoff data for SPRNT to input is m3/sec.

4.3 Runoff Netcdf file

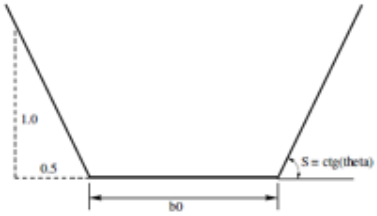
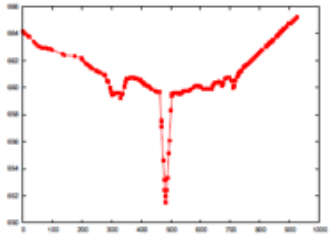
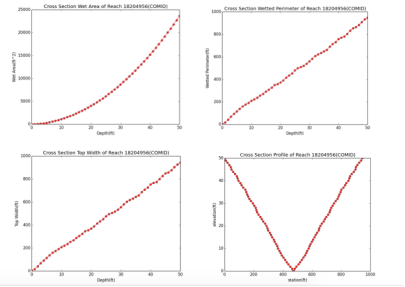
In large-scale hydraulic modeling, sometimes we have to deal with excessive runoff data that can up to several gigabytes. The efficiently way to handle these data is saving them as a NetCDF file. NetCDF file is a file type commonly used in climatology, meteorology and oceanography applications. It is also a common file type for GIS tools. More information about NetCDF file: `https://en.wikipedia.org/wiki/NetCDF` In SPRNT simulation, the runoff data has to be save into a NetCDF file in the specific format. The example netcdf file in */SPRNT/Tutorial/ExampleNetcdf/* can be used for users' reference.

4.4 NHDPlus flowline

The SPRNT model uses the NHD Flowline as the frame of the river network. NHD Flowline features can be download from the NHDPlus version 2 website `http://www.horizon-systems.com/NHDPlus`.

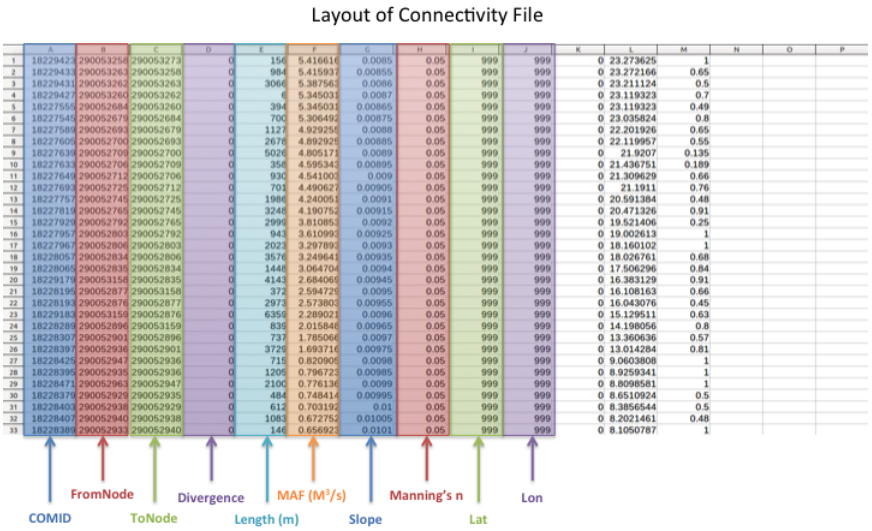
4.5 River geometry

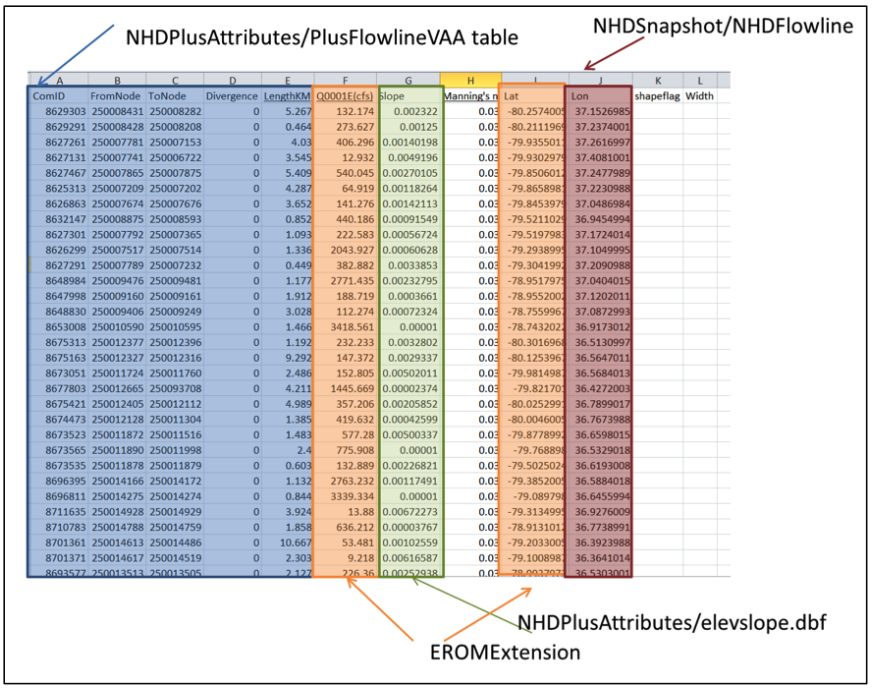
River geometry is the important input factors in hydraulic routing models. In SPRNT model, there are several types of cross section descriptions are supported for users' choice:

Types of cross section description	Required Variables
Trapezoidal: Two parameters needed to be defined: bottom width and sidewall slope.	
XY Coordinate system: Need **X (station)** and **Y (elevation)** coordinates to describe the channel shape.	
Intrinsic (Hydraulic Propertis): Unlike traditional hydraulic models, SPRNT provides a new way to describe channel shape - "Intrinsic cross section description". In intrinsic description, user will be asked to provide certain hydraulic properties in a certain order: **A-P-Y-W**. Where A is wetted area, P is wetted perimeter, Y is water depth and W is top width.	

4.6 Connectivity file

The figure below illustrates the format of connectivity file. You can find colored columns from each denoted file from NHDPlus data:
`http://www.horizon-systems.com/NHDPlus/NHDPlusV2_data.php`



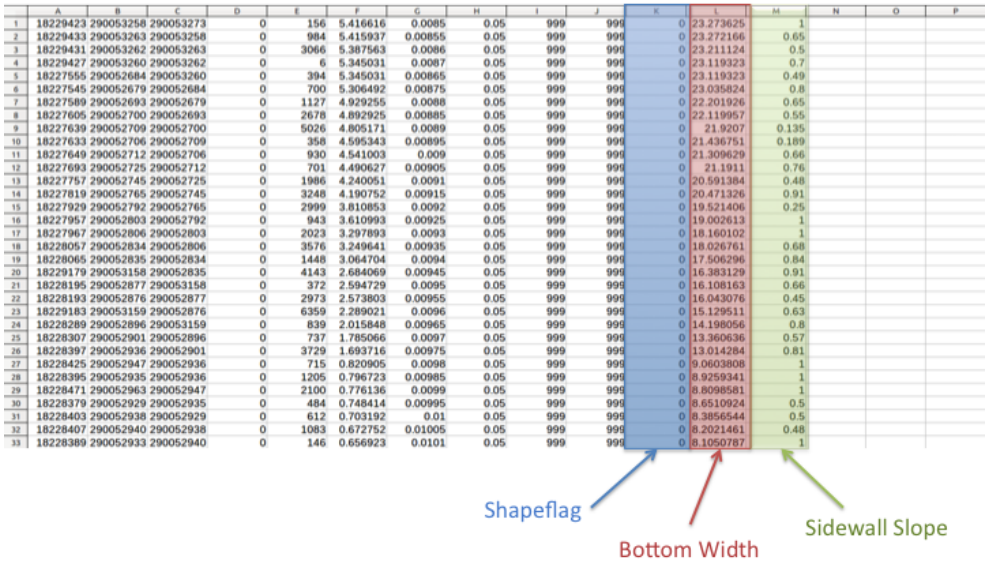


ComID	FromNode	ToNode	Divergence	LengthKM	Q0001E(cfs)	Slope	Manning's n	Lat	Lon	shapeflag	Width
8629303	250008431	250008282	0	5.267	132.174	0.002322	0.03	-80.2574005	37.1526985		
8629291	250008428	250008208	0	0.464	273.627	0.00125	0.03	-80.2111969	37.2374001		
8627261	250007781	250007153	0	4.03	406.296	0.00140198	0.03	-79.9355011	37.2616997		
8627131	250007741	250006722	0	3.545	12.932	0.0049196	0.03	-79.9302979	37.4081001		
8627467	250007865	250007875	0	5.409	540.045	0.00270105	0.03	-79.8506012	37.2477989		
8625313	250007209	250007202	0	4.287	64.919	0.00118264	0.03	-79.8658981	37.2230988		
8626863	250007674	250007676	0	3.652	141.276	0.00142113	0.03	-79.8453979	37.0486984		
8632147	250008875	250008593	0	0.852	440.186	0.00091549	0.03	-79.5211029	36.9454994		
8627301	250007792	250007365	0	1.093	222.583	0.00056724	0.03	-79.5197983	37.1724014		
8626299	250007517	250007514	0	1.336	2043.927	0.00060628	0.03	-79.2938995	37.1049995		
8627291	250007789	250007232	0	0.449	382.882	0.0033853	0.03	-79.3041992	37.2090988		
8648984	250009476	250009481	0	1.177	2771.435	0.00232795	0.03	-78.9517975	37.0404015		
8647998	250009160	250009161	0	1.912	188.719	0.0003661	0.03	-78.9552002	37.1202011		
8648830	250009406	250009249	0	3.028	112.274	0.00072324	0.03	-78.7559967	37.0872993		
8653008	250010590	250010595	0	1.466	3418.561	0.00001	0.03	-78.7432022	36.9173012		
8675313	250012377	250012396	0	1.192	232.233	0.0032802	0.03	-80.3016968	36.5130997		
8675163	250012327	250012316	0	9.292	147.372	0.0029337	0.03	-80.1253967	36.5647011		
8673051	250011724	250011760	0	2.486	152.805	0.00502011	0.03	-79.9814981	36.5684013		
8677803	250012665	250093708	0	4.211	1445.669	0.00002374	0.03	-79.821701	36.4272003		
8675421	250012405	250012112	0	4.989	357.206	0.00205852	0.03	-80.0252991	36.7899017		
8674473	250012128	250011304	0	1.385	419.632	0.00042599	0.03	-80.0046005	36.7673988		
8673523	250011872	250011516	0	1.483	577.28	0.00500337	0.03	-79.8778992	36.6598015		
8673565	250011890	250011998	0	2.4	775.908	0.00001	0.03	-79.768898	36.5329018		
8673535	250011878	250011879	0	0.603	132.889	0.00226821	0.03	-79.5025024	36.6193008		
8696395	250014166	250014172	0	1.132	2763.232	0.00117491	0.03	-79.3852009	36.5884018		
8696811	250014275	250014274	0	0.844	3339.334	0.00001	0.03	-79.089798	36.6455994		
8711635	250014928	250014929	0	3.924	13.88	0.00672273	0.03	-79.3134995	36.9276009		
8710783	250014788	250014759	0	1.858	636.212	0.00003767	0.03	-78.9131012	36.7738991		
8701361	250014613	250014486	0	10.667	53.481	0.00102559	0.03	-79.2033005	36.3923988		
8701371	250014617	250014519	0	2.303	9.218	0.00616587	0.03	-79.1008981	36.3641014		
8693577	250013513	250013505	0	2.127	226.36	0.00252938	0.03	-78.9037971	36.5303001		

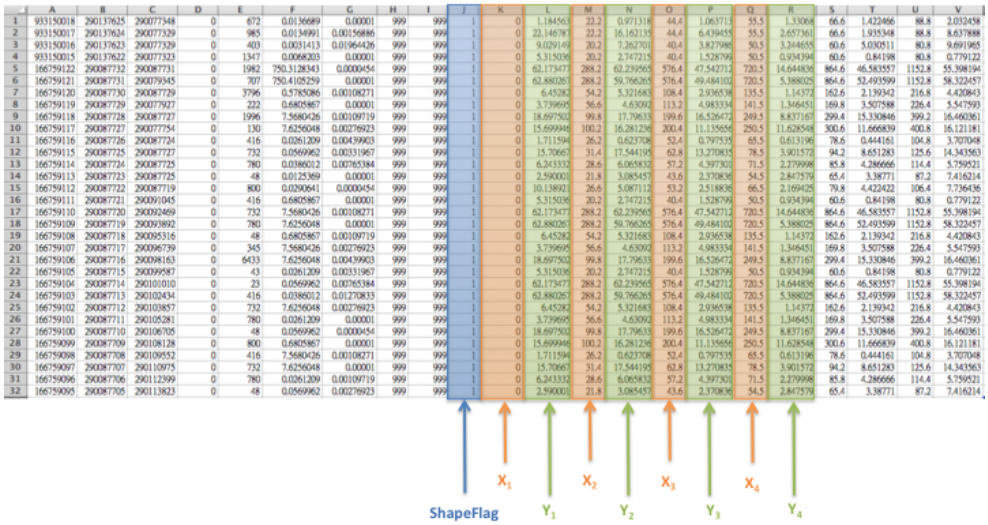
For Manning's n, users will need to specify the value between 0.03 - 0.05 and compare the result with USGS data. Shapeflag is an indicator that represents the types of cross-sections. It can be assigned by three values. (0 : Trapezoidal, 1: XY coordinate and 2: Intrinsic (hydraulic properties)). When the Shapeflag equals to 0 (trapezoidal description), the following two columns will be Bottom Width and Side Wall Slope. If the Shapeflag was assigned to 1 (XY coordinate), user will need to put the XY data after the Shapeflag column in the order of X_1, Y_1, X_2, Y_2...X_n, Y_n.

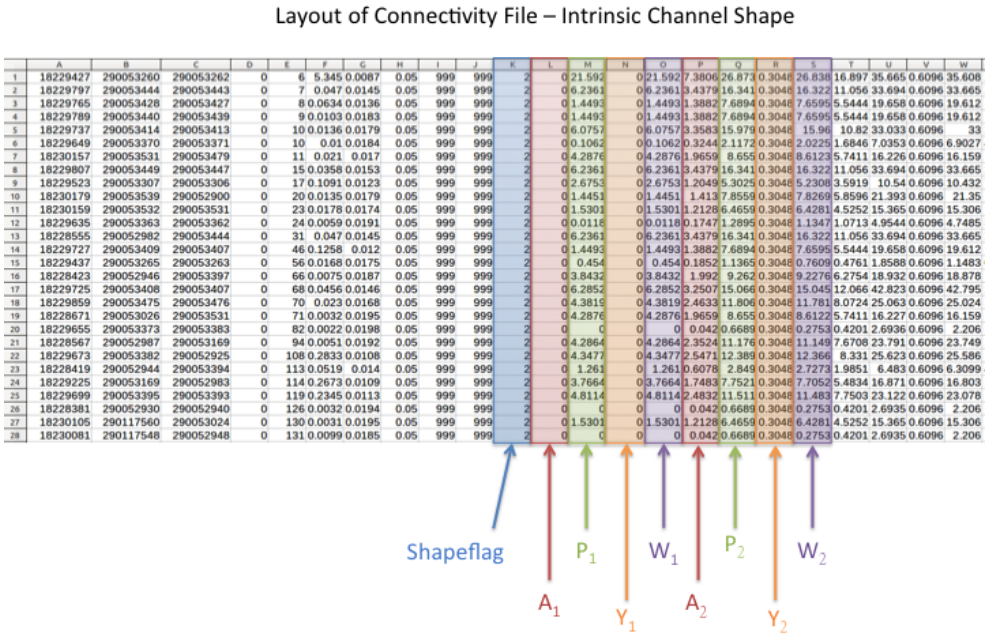
If user assign the Shapeflag to 2 (Intrinsic description), hydraulic properties should be put in the order of A_1, P_1, Y_1, W_1, A_2, P_2, Y_2, W_2,..... A_n, P_n, Y_n, W_n. In order to let users to have a better knowledge, two types of connectivity file by using different types of cross section description were illustrated in the following figures:

Layout of Connectivity File – Trapezoidal Channel Shape



Layout of Connectivity File – XY Coordinate Channel Shape





4.7 Netlist file (.spt) structure

"Netlist" is an idea that previously used to describe electric circuit topography in computer science. SPRNT model uses the netlist file, which is a text-based file includes a series combinations of "def" and "end", to define the channel geometric characteristics, forcing terms, junction contribution ratios and other factors of the simulation.

In here, for users' convenience, we provide a ready-to-use preprocessor(written in C++) for producing netlist files. Please refer to **"HOW TO USE PREPROCESSOR"** section in appendix for detailed instructions. The following figure is a small sample of a SPRNT netlist excerpts from Liu and Hodges (2014). For more details on the SPRNT netlist, please refer to the SPRNT users' manual written by Frank Liu (2012). `https://github.com/frank-y-liu/SPRNT/blob/master/doc/sprint_manual_rev.pdf`

4.8 Execute SPRNT model commands

Once the netlist file was done (the example here is Davis_Creek.spt), move the netlist file to the bin directory by using the following commands:

```
$ mv FILE_NAME.spt WHERE_YOU_SAVE_SPRNT/SPRNT/bin/
```

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ mv Davis_Creek.spt ~/Desktop/Github_SPRNT/SPRNT/bin/
```

Then we have to move to the /SPRNT/bin/ directory by using cd command:

```
$ cd WHERE_YOU_SAVE_SPRNT/SPRNT/bin/
```

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ cd ~/Desktop/Github_SPRNT/SPRNT/bin/
```

Now, your working directory should be /SPRNT/bin. You can use the command "ls" to list all the visible files in this directory. The netlist file we just moved to /bin/ directory should be here as well (example file Davis_Creek.spt)

```
$ ls
```

```
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ ls
chk_output.sh  Davis_Creek.spt  extract_node.sh  extract_time.sh  run_spt.sh  sprnt
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$
```

To start the SPRNT simulation, we have to execute a shell script called "run_spt.sh".

```
$ sh run_spt.sh FILENAME.spt
```

```
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ ls
chk_output.sh  Davis_Creek.spt  extract_node.sh  extract_time.sh  run_spt.sh  sprnt
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ sh run_spt.sh Davis_Creek.spt
```

If the SPRNT model is successfully launched, user should see the following message:

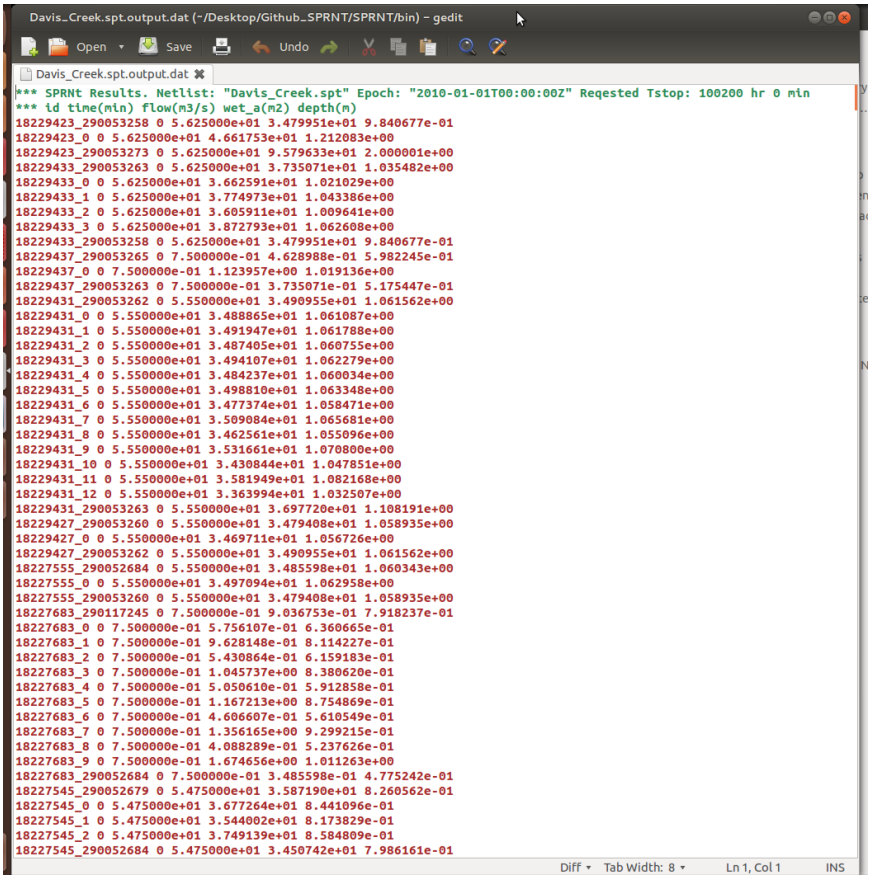
```
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ ls
chk_output.sh  Davis_Creek.spt  extract_node.sh  extract_time.sh  run_spt.sh  sprnt
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ sh run_spt.sh Davis_Creek.spt

   SPRNT: Simulation Program for River Networks
      version 1.3.1 (73a56f89) : Sep 18 2015  13:52:54
      compiled with 4.6.3

[II]: Found 1631 nodes, 1400 segmetns, 156 junctions, 75 q_sources, 697 lateral sources, 1 bdn condtion
he netlist
[II]: Metric set to 1
[II]: Verbose set to 1
[II]: PrtDepth set to 1
[II]: PrtQ set to 1
[II]: PrtA set to 1
[II]: SSFile (SteadyState file) set to "Davis.ssf"
[II]: Epoch is set to 2010-01-01T00:00:00Z
[II]: PrtInterval set to 12000 min
[II]: StopTime set to 3.607e+08 second
[II]: successfully allocated solver, type = 1, size = 3262
[II]: Steady state solve converged in 12 steps
  S Iter w/ dt=2.50e+01 passed, nxt_dt=2.50e+01, num_iter=  2, max_norm=2.31e-05, tdif=3.02e-06, n_sc=
  S Iter w/ dt=2.50e+01 passed, nxt_dt=2.50e+01, num_iter=  1, max_norm=2.44e-06, tdif=1.05e-06, n_sc=
  S Iter w/ dt=2.50e+01 passed, nxt_dt=2.50e+01, num_iter=  1, max_norm=1.80e-06, tdif=4.76e-07, n_sc=
  S Iter w/ dt=2.50e+01 passed, nxt_dt=2.50e+01, num_iter=  0, max_norm=9.80e-07, tdif=0.00e+00, n_sc=
[II]: Number of nodes = 1631, size of matrix = 3262
[II]: SteadyState phase II converged in 4 steps with 4 iterations.
[II]: Saved steady state to file "Davis.ssf"
[II]: Unsteady results will be stored in "Davis_Creek.spt.output.dat"
  U time=0.0000e+00 (  0.00 out of 100200.00 hrs), num_iter= 0, falter=0, dt=1.50e+01, n_sc=145, max_f=
n_mn=  0
  U time=1.5000e+01 (  0.00 out of 100200.00 hrs), num_iter= 0, falter=0, dt=1.50e+01, n_sc=145, max_f=
n_mn=  0
  U time=3.0000e+01 (  0.01 out of 100200.00 hrs), num_iter= 0, falter=0, dt=1.50e+01, n_sc=145, max_f=
n_mn=  0
  U time=4.5000e+01 (  0.01 out of 100200.00 hrs), num_iter= 0, falter=0, dt=1.50e+01, n_sc=145, max_f=
n_mn=  0
  U time=6.0000e+01 (  0.02 out of 100200.00 hrs), num_iter= 0, falter=0, dt=1.50e+01, n_sc=145, max_f=
n_mn=  0
  U time=7.5000e+01 (  0.02 out of 100200.00 hrs), num_iter= 1, falter=0, dt=3.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=1.0500e+02 (  0.03 out of 100200.00 hrs), num_iter= 0, falter=0, dt=3.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=1.3500e+02 (  0.04 out of 100200.00 hrs), num_iter= 0, falter=0, dt=3.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=1.6500e+02 (  0.05 out of 100200.00 hrs), num_iter= 0, falter=0, dt=3.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=1.9500e+02 (  0.05 out of 100200.00 hrs), num_iter= 0, falter=0, dt=3.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=2.2500e+02 (  0.06 out of 100200.00 hrs), num_iter= 1, falter=0, dt=6.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=2.8500e+02 (  0.08 out of 100200.00 hrs), num_iter= 1, falter=0, dt=6.00e+01, n_sc=145, max_f=
n_mn=  0
  U time=3.4500e+02 (  0.10 out of 100200.00 hrs), num_iter= 0, falter=0, dt=6.00e+01, n_sc=145, max_f=
n_mn=  0
```

When the execution is successful, the model will first solve the steady state convergence. After the steady state successfully converges, the steady state result will be saved to a FILENAME.ssf file. The ".ssf" file was not designed for users to read, so there will After the .ssf file was created, SPRNT model will start solving the unsteady state convergence. The unsteady state results will be saved to a FILENAME.spt.output.dat file.

```
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$ ls
chk_output.sh     Davis_Creek.spt.output.dat  extract_node.sh  run_spt.sh
Davis_Creek.spt   Davis.ssf                   extract_time.sh  sprnt
yuchanway@Hodges-MacPro:~/Desktop/Github_SPRNT/SPRNT/bin$
```

4.9 Result file structure

After the unsteady state convergence finished, you will be able to see the file FILENAME.spt.output.dat in your /bin/ directory. The following screen snapshot is the output file of SPRNT model, named "Davis_Creek.spt.output.dat". Basically, the output file is a space-delimitated text-based file, so it can be open by any kind of text reader.



4.10 "Checkonly" Mode in SPRNT

In here, SPRNT also provides a "Check-only" function for users to do quick topological check, which will print out the general information at each computation point, including Flowrate, Wetted Area, Water Depth and Froude number. This check mode can be executed by using the following command:

```
$ sh run_spt.sh -checkonly FILENAME.spt
```

5 FAQ for SPRNT

In this section, some frequently asked questions and common error messages are listed for users reference. Every effort has been made to make this FAQ as informative as possible; if you have any suggestions as to how it may be improved, send them to Cheng-Wei Yu (`yuchanway@utexas.edu`).

1. **[EE] Bummer: time step too small at time point XXX second.**
   This error message is caused by the oscillation of the numerical solution, which is known as "Gibb's phenomenon". Usually, Gibb's phenomenon will occur before a huge, flashy and suddenly runoff event. There are several appropriate solutions for this problem, and the most useful one is applying a filter (ex. Gaussian filter, lowpass filter) to smooth the peak and approach to the real hydrograph.

2. **Bummer: wetted area specification in INTRINSIC x-section has to be monotonically increasing.**
   Before running a SPRNT simulation, SPRNT will do a simple topography and river geometry check. If the user uses the intrinsic cross section to describe the channel property, the wetted area should increase with the depth increase. If SPRNT finds any of the channel properties violate this law, it will show this error message.

3. **[EE]: first phase of steady-state solve failed.**
   There are several possibilities to cause the steady state convergence failure, the most common one is the missetting of boundary conditions (ex. downstream boundary or contribution ratio in the junction).

4. **[EE]: second phase of steady state solve failed.**
   The reason to have this error message is similar to the "first phase steady state solve failed". Once have this error message, please check your boundary conditions and other settings.

5. **DBINTK error**
   This message only shows when you have over-flat channel sidewalls. Since SPRNT is a one-dimensional hydrodynamic model, there are some limitations for the channel sidewall slope. The minimum sidewall slope value that SPRNT can take is 0.0015 (this value can vary due to different channel characteristics), once the channel sidewall slope is too flat, the flow in this channel section in the simulation will become a two-dimensional problem, which cannot be solved by SPRNT model.

6. **Bummer: error reading 4D data specified in line XX, requires at least 5 points.**
   The default requirement for an intrinsic cross section is 5 sets (A, P, Y, W) of data. If the given cross section data is less than 5 sets, SPRNT model will not be bale to build a complete cross section shape.

7. **== tchk == node "XXXX_XXXX" requires upstreaming Q.**
   The upstream forcing term cannot be set to 0.
   Unlike hydrological models, the upstream boundary flows can not be set to 0 in hydrodynamic models due to the necessity of initial state and prevention of singularity matrices in numerical simulation.

8. **== tchk == node "XXXX_XXXX" no flow path to downstream.**
   Two common reasons to have this error message. (1) Unbalance contribution ratios in the junction point. When there is a great difference between two inflow upstream in the junction point, flow from the minor upstream will be blocked and the flowline will be regarded as "no flow path". For this case, users need to check if contribution ratios in junction are reasonable. (2) Unreasonable channel cross section combination. Overall, in natural rivers, the area of river bathymetry will gradually increase with the cumulative catchment area/flowrate, and this rule also applies to SPRNT model cross section setting. When users use the unreasonable cross section setting, for instance, wide upstream and narrow-shallow downstream, this error message will be triggered.

Appendices

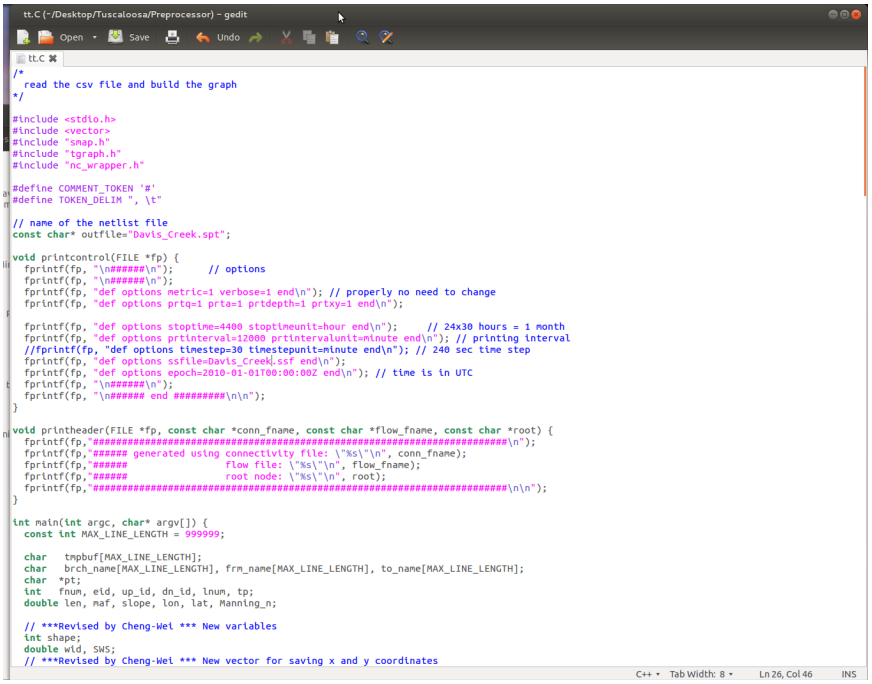
A How to use the Preprocessor

In here, we provided a ready-to-use data preprocessor for users' convenience to make a proper netlist files for running SPRNT model. In this section, we will go through every step of how to set up the simulation variables such as upstream/downstream boundary conditions, computational time-steps, result print time steps, and output filename in this ready-to-use preprocessor.

[NOTE]: Mac users may need to install NetCDF library through Xcode when compiling the source code of this preprocessor, detailed tutorial about installing NetCDF on Mac machine can be found on the Internet.

Set up output filename and simulation time

In the preprocessor directory, users can easily find a file called *"tt.C"*, which is the environmental variables set up file of the SPRNT model. *tt.C* also contains the main function of the preprocessor.
In this file, we will be able to configure the output filename (the *.spt file that we will use for running SPRNT), the steady state result filename, the length of simulation time, the length of computational time steps, and the printing interval in the SPRNT result file. The following picture is the screen snapshot of tt.c.



```
char* outfile = "Davis_Creek.spt"
```

The "Davis_Creek.spt" here is the output filename you will produce from the preprocessor. It can be changed to any name end with .spt.

```
fprintf(fp, "def options stoptime=4400 stoptimeunit=hour end\n"
```

The total length of simulation time can be set up in the above code line, in the example shown here, the simulation length is 4400 hours, which is about half year time. Please be aware of that the total simulation time should correspond to the time series you have in the *Runoff NetCDF file*, or it can be shorter if users just need to simulate a particular time period.

```
fprintf(fp, "def options prtinterval=12000 prtintervalunit=minute end\n");
```

The command line here can control the printing interval in the result file, which is a "*.spt.output.dat" file (users can go to section **"3.9 Result file structure"** for more details). Please notice that the printing interval unit is minutes. Since the default runoff time step from land surface model is normally hourly, the author suggests that the printing interval should be no shorter than 60 minutes. The finer of the printing interval, the bigger result file you will have. Sometimes the result file will up to > 100GB when simulating a long simulation with fine printing intervals. The oversized output file will enhance the difficulty for users to open/read the result in the file, so choosing optimal printing interval wisely is a crucial task for users to decide.

```
fprintf(fp, "def options timestep=30 timestepunit=minute end\n");
```

This code line decides the length of computation time steps in the SPRNT model. However, the convergence stability of fixed length computational timestep in the finite difference method has been addressed in numbers of articles. Users need to use caution when setting a fixed computational timestep in this line. Fortunately, the SPRNT model provides a dynamic time step function, which can adjust the computational length automatically. If users want the SPRNT use the dynamic timestep function, just simply comment out this line (by adding "//" at the beginning of the line).

```
fprintf(fp, "def options ssfile=Davis_Creek.ssf end\n");
```

In the SPRNT model, the steady state simulation will be solved before solving unsteady state computation. The result of the steady state will be saved in a " *.ssf " file and used as the initial condition of unsteady state computation. In here, we use **Davis_Creek.ssf** as an example.
*[NOTE]: *.ssf is a text-base file for temporary data storage, and it is not a*

*well-formatted output file for recording results. Users can disregard the contents of the *.ssf file.*

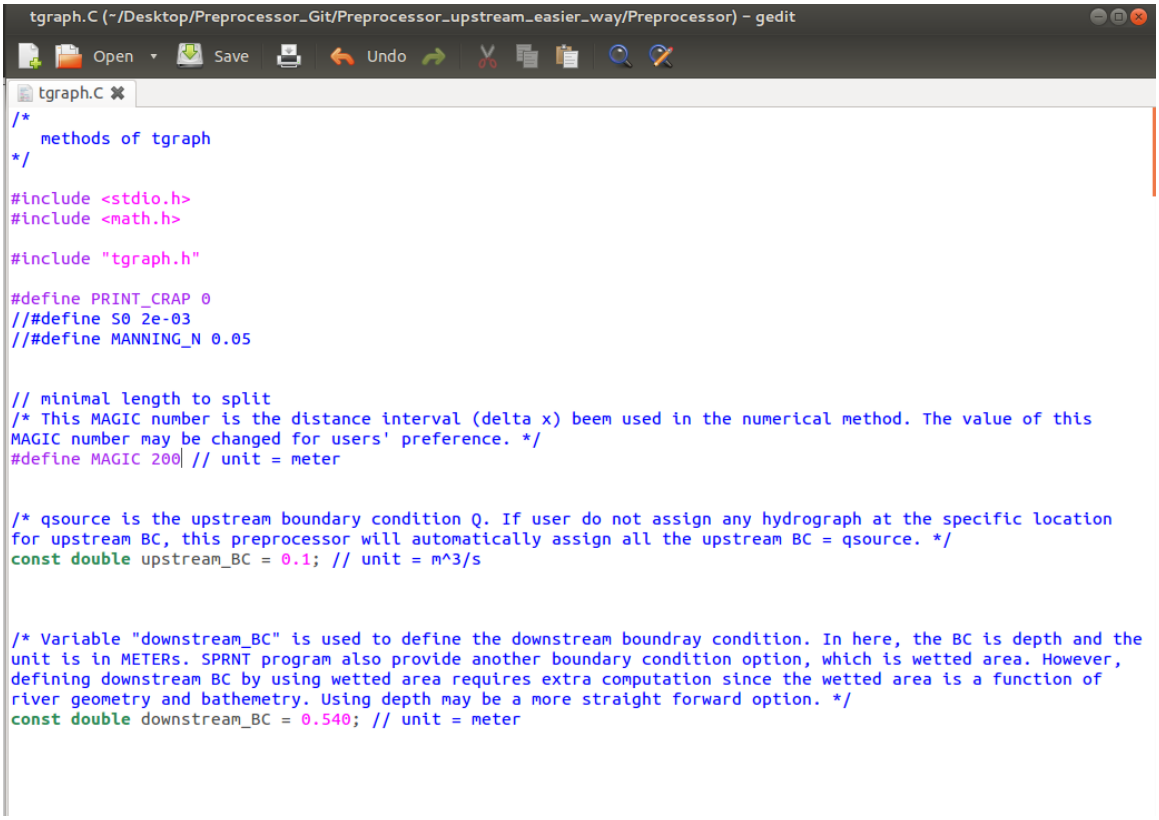
```
fprintf(fp, "def options epoch=2010-01-01T00:00:00Z end\n");
```

The starting UTC date and time (Users need to convert the time to appropriate time zone corresponds to the study area) of simulation can be set up here. The time setting here won't have any influence on the model result, it will only be printed in the header of the result file for users' reference.

Set up boundary conditions

Boundary conditions can be defined in *"tgraph.C"* file, which includes several different C++ classes to control the *def - end* block printing in the netlist file. In this file, we will be able to define the downstream BC, upstream BC (forcing terms) and the length of spatial computational nodes.

At the top of the *tgrapg.C* file, users can find the codes as shown in the following figure:



In here, users can define boundary conditions and spatial length of computational nodes. Detailed instructions are elaborated in the following:

Spatial interval of computational node

```
#define MAGIC 200
```

This code line defines the spatial length of the computational node in the river network. In here, we use 200 meters as an instance, which means every flowline longer than 200 meters will be divided into several segments and computational nodes will be inserted between segments. For the flowline is shorter than 200 meters, the preprocessor will define it as 200 meters.
[NOTE]: When setting the spatial interval length, Courant stability condition need to be contemplated by users in order to get a stable numerical solution.

Upstream boundary condition (forcing term)

```
const double upstream_BC = 0.1;
```

In hydraulic/hydrodynamic models, the upstream boundary conditions are required to create an initial solution to avoid St. Venant Equations become singular. Normally, upstream boundary conditions(forcing terms) are often referred as the river "baseflow", which can be found from the public/private agent such as USGS website. Recommended value for forcing terms is between 0.02 m^3/s to 0.1 m^3/s.

Downstream boundary condition

```
const double downstream_BC = 0.540;
```

For downstream boundary condition, users need to specify the water depth (in meters) for the outlet point of the river network.

How to compile the preprocessor

After setting up all the variables, users can compile the preprocessor by using following commands:

```
$ make
```

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ make
g++ -Wall -std=c++0x -O3   -c -o nc_wrapper.o nc_wrapper.C
g++ -Wall -std=c++0x -O3   -c -o tgraph.o tgraph.C
tgraph.C: In member function 'int tgraph::DFSupstream(const char*, NC*, FILE*)':
tgraph.C:244:15: warning: variable 'cntr' set but not used [-Wunused-but-set-variable]
g++ -Wall -std=c++0x -O3   -c -o tt.o tt.C
tt.C: In function 'int main(int, char**)':
tt.C:190:42: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
tt.C:196:44: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
g++ nc_wrapper.o tgraph.o tt.o -o a.out -O3 -L/usr/local/lib -lnetcdf
```

After executing the *"make"* command, users will see an executable file called **"a.out"** in the directory. This executable file is an assembler output file for connecting all the required codes/libraries to an executable code. Users can use this executable object to load the data in connectivity file and runoff NetCDF file by using the following commands:

```
$ a.out Connectivity_filename.csv Runoff_NetCDF.nc
```

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ ./a.out Davis_Creek_composite.csv Tuscaloosa_RC.nc
```

In this step, the preprocessor will scan all the data in the connectivity file(*.csv) and runoff NetCDF file(*.nc), to verify that the topography data matches the runoff data. Once the examination is done, the preprocessor will return the number of river network system in the study area by denoting the outlet point "ToNode":

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ ./a.out Davis_Creek_composite.csv Tuscaloosa_RC.nc
Total 232 nodes.
Fixed 49 edges with length less than 222.0
Region 0, root = 290053273, num vertices = 232
Reading runoff data from Tuscaloosa_RC.nc
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$
```

In our example here, we have only one river network in the Davis_Creek_composite.csv connectivity file, with outlet point "290053273". If user wants to proceed to produce the netlist file, further command is required:

```
$ a.out Connectivity_filename.csv Runoff_NetCDF.nc Node_Number
```

```
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ ./a.out Davis_Creek_composite.csv Tuscaloosa_RC.nc
Total 232 nodes.
Fixed 49 edges with length less than 222.0
Region 0, root = 290053273, num vertices = 232
Reading runoff data from Tuscaloosa_RC.nc
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$ ./a.out Davis_Creek_composite.csv Tuscaloosa_RC.nc 290053273
Total 232 nodes.
Fixed 49 edges with length less than 222.0
Region 0, root = 290053273, num vertices = 232
Reading runoff data from Tuscaloosa_RC.nc
netlist will be saved as Davis_Creek.spt
Traversing upstream from root 290053273
Total Lateral = 0.000e+00, Total Q = 3.750e+01
yuchanway@Hodges-MacPro:~/Desktop/Tuscaloosa/Preprocessor$
```

After executing above commands, the preprocessor will start to discard useless data (if there are other networks in the connectivity file) and extract the designate river network to produce the netlist file(*.spt). The final netlist filename will be the name that users setup in the **"tt.C"** file.

The final netlist file will be moved to the bin directory in SPRNT folder for running the simulation.

Re-compile the preprocessor

If there is any change in variables, please make sure to recompile the preprocessor by using the following two commands:

```
$ make clean
```

```
$ make realclean
```

These two commands will clean up all the old executable objects and dependency files. After setting up the new conditions/variables, remember to do the **"make"** command again to create a new executable object.