Expert-N 5.0 Manual

Christian Klein

04.12.2013

Contents

Chapter 1

Overview

1.1 Introduction

Expert-N 5.0 is a simulation framework, which provides an interface to manage and combine dynamic modules to a complete model. In comparison to former versions of Expert-N (Stenger et al., 1999) number, order and configuration of modules and sub-modules can be defined almost free. This is obtained by configuration files and a plug-in like structure, which loads module packages during run time. The flexibility makes Expert-N 5.0 to a workbench that is ready for many scientific questions and the related modules, which are needed in future can be easily included into the framework. However, it makes sense to choose a proven combination of modules and sub-modules for the start configuration to reduce the complexity of the system and limit the debugging time. The combination of modules for water, heat, nitrogen and carbon dynamics, which was chosen here, is an example for a soil-plant ecosystem simulator, which can be later completed with other modules like pollen forecast, animal models, decision support systems, etc. The technical details and the configuration process is explained in Chapter 3.
Another new property is that the Expert-N 5.0 core is designed as library itself. This makes it possible to couple Expert-N to any application or model. For example, it is possible to realize an Expert-N Version which works as a stand alone version like former versions of Expert-N, or to connect Expert-N to a weather and forecast model including two-way interaction. At the present time two drivers exist: The first is a stand alone version, which use climate data as input files, and the second is a version which is coupled to the Weather Research and Forecasting (WRF) model Skamarock et al. (2008). Details to driver implementations are explained in Section 4.1
Communication between driver and modules can be done in two ways: First, there is an implemented base structure, a predefined structure with plant, soil, heat and water variables, which satisfies the communication for the most plant and soil models. The second way of communication works with an associative array, which gives the modules the possibility to share structures between models, between driver and models or even between parallel instances of Expert-N. More details are explained in Section 4.2.
The supported algorithms of the modules are taken from such published models like LEACHM (Hutson and Wagenet, 1992), HYDRUS (Simuunek et al., 1998), N-SIM (Schaaf et al., 1995),Modeling Plant and Soil (Horton et al., 1991), NCSOIL (Nicolardot and Molina, 1994), Hurley Pasture Model (Thornley et al., 1998), NOAH (Chen and Dudhia, 2001), CENTURY (Parton et al., 1998) and DAISY (Hansen et al., 1991).
Simulations are preformed one-dimensionally in the vertical direction and, depending on the spatial variation of main parameters, considered as representative for the plot or the field scale. It is possible to arrange several Expert-N plant-soil columns in a gridded structure. The grid can be configured to match the Arakawa-C grid of WRF (Weather and Research Model) Skamarock et al. (2008). Additionally it is possible to configure several numbers of Expert-N plant-soil columns on every grid cell. Owing to its plugin-like structure, there are no fixed modules in Expert-N 5.0. Every module can be displaced or removed. However, there are a few modules which take a leading part in memory management, input and output of data and are recommended to be used in every session.

1.2 Expert-N 5.0 Modeling System Program Components

To use the tools of Expert-N or the Expert-N Standalone version it is is necessary to setup the Expert-N Environment (–> Section 2.3.1).

1.2.1 Expert-N 5.0 standalone version:

- **expertn_bin** or **expertn_bin.exe**: located in the **expertn50/built/bin path** and is the heart of the Expert-N Modeling System

1.2.2 Expert-N Tools:

- expertn_gui.py
- xnoplot.py
- xpnhelper.py
- run_n_times.py
- xpn_cfg_manager.py
- dump_wrfnc.py
- transpose_cdb_file.py
- xpnstatistic.py
- create_input_for_rosetta_from_xpn.py
- convert_rosetta_to_xpn.py
- find_grid_koord.py
- calc_statistic.py
- create_project_from_netcdf.py

Chapter 2

Software Installation

2.1 Introduction

2.2 Required Compilers, Scripting Languages and Libraries

2.2.1 Architecture

Expert-N was tested with the following operating systems and architectures (it might work with other software and architecture, too):

- Suse Linux Enterprise Server 11 (Kernel-Release: 2.6.32.12-0.7, Kernel-Version: #1 SMP 2010-05-20 11:14:20, Architecture: x86_64, Operating System: GNU/Linux)
- Ubuntu 12.04 (Architecture: i386_64 bit)
- Windows XP Professional (Architecture: i386_32 bit), Windows 7

2.2.2 Software Requirements

Most libraries and tools are available in the packet manager of any linux distribution. In principle, it is possible to use any POSIX C compatible compiler (on not POSIX compatible systems, glibc functions are used, if possible). On non POSIX systems, like MS Windows, some Expert-N 5.0 functions are not available.

Expert-N Console:

- Python 2.7 (for waf build tool)
- gcc, g++ (compiler), Visual C++ (Windows)
- GLib 2.22.5 (lib)
- GObject 2.22.5 (lib)
- GModule (lib)
- libxml (lib)
- GDA 4.1.2 (lib)
- NETCDF (lib)
- PNETCDF (used by OPEN MPI)
- OPEN MPI (optional)

Expert-N GUI (configuration tool):

- Python 2.7
- Bash Interpreter
- PyGTK 2.16,
- GTK 2.18
- Numpy 1.3
- Matplotlib 0.99.1.2
- RSVG 2.26
- python-rsvg 2.30
- Text Editor, Office

2.3 Building Expert-N

2.3.1 Building on Ubuntu 12.04 with OpenMP or MPI

Checkout:

- Install Subversion (svn):

```
sudo apt-get install subversion
```

- Checkout (e.g. in your Home (~) path):
  - get **expertn50** and **expertn50/dev** path:

```
svn checkout http://newmatrix/projects/expertn50
```

  - change to the **expertn50** path and checkout the **built** and the **expertn_tools** path:

```
cd expertn50
svn checkout http://newmatrix/projects/built
svn checkout http://newmatrix/projects/expertn_tools
```

  - the expertn_gui and xnoplot need cfg files (there are some templates, you can modify them later to your needs) –> type:

```
cd expertn50/expertn_tools
cp expertn_gui.ini.template expertn_gui.ini
cp xnoplot.ini.template xnoplot.ini
```

Installing Software Packages:

- Open a terminal and type:

```
sudo apt-get install libgtk2.0-dev libgda-4.0-dev libnetcdf-dev
```

- For the configuration tool scripts:

```
sudo apt-get install python-numpy python-matplotlib python-rsvg python-
   iniparse
```

- If you want to use OpenMP parallelization you can go to the configure step.
- If you want to use MPI parallelization you have to install the MPI Software:
  - ```
    sudo apt-get install libopenmpi-dev openmpi-bin
    ```
  - Expert-N uses library pnetcdf for netcdf output, in case mpi is used. This library is not available in Ubuntu's packet manager. Download it from: `http://trac.mcs.anl.gov/projects/parallel-netcdf`
  - Download for example Version 1.4.0 (File: parallel-netcdf-1.4.0.tar.gz)
  - Extract it into a directory of your choice and install it with:

```
export CPPFLAGS=-fPIC
./configure --disable-fortran
make
sudo make install
```

Configure, Build and Install Expert-N:

- move to dev path in your Expert-N directory, e.g:

```
cd ~/expertn50/dev
```

- check the possible installation and configuration possibilities:

```
./waf --help
```

- configure process (take your preferred parallelization option)
  - with OPENMP and optimization:

```
./waf configure --USE_OPENMP --max_optimize
```
```

– with MPI and optimization:

```
./waf configure --USE_MPI --max_optimize
```

- if everything is OK, just type:

```
./waf build install
```

and your Expert-N should be compiled and installed to the built directories. e.g: ~/expertn50/built

Checking your installation

- setup the Expert-N Environment:

```
export XPN_PATH=~/expertn50
~/expertn50/built/bin/expertn_environment.sh
```

You can add this expression to your ~/.bashrc file to setup the Expert-N Environment each time you open a bash terminal

- type something like:

```
expertn_bin --version
```

or

```
expertn_bin --help
```

2.3.2 Installing additional Software (optional)

This software is not necessary to use Expert-N. It is only my recommendation!

- To use the netcdf access of the python scripts in the **expertn_tools** path, the library **netcdf4python** is necessary:
  – Download from: `http://code.google.com/p/netcdf4-python/`, e.g: netCDF4-1.0.7.tar.gz
  – Type:

```
sudo apt-get install libhdf5-serial-dev python2.7-dev
sudo ldconfig
sudo python setup.py install
```

- Text Editor (Geany: Lightweight text editor, also useful for programming, scripts and small projects):

```
sudo apt-get install geany geany-plugins
```

- SQLite Manager for Databases:

```
sudo apt-get install sqliteman
```

- Integrated Development Environment (IDE) (Codelite): `http://codelite.org/`

2.3.3 Compiling an example module

Expert-N 5.0 is very flexible. It possible to create new modules and models in C and C++ easily. There is an example project in **dev/xpn/modul/libmodule_example/src**. It consists of the following files:

file	description
wscript	configuration file for the waf build system (dependencies, libname, target, project file names, ...)
module_example_register_modul.c	Expert-N description file for the plugin: This plugin consists two models (biomass growth and photosynthesis)
module_example.h	header file for an example gobject class
module_example.c	model example

Configure Process with the waf build system:

- goto to the directory **dev**
- There you find the file **module_example.mod**, a text file with the path to your example project
- So let's configure the project:

```
./waf configure --debug --USE_OPENMP --extern_project=module_example.mod --
   prefix=$XPN_PATH/built
```

  - –debug: compile in debugging mode (alternative you can use –max_optimize for maximum of speed:
  - –USE_OPENMP: OPENMP parallelization (alternative: USE_MPI)
  - –extern_project: text file with paths to the modules, which will be compiled
  - –prefix: Expert-N 5.0 installation path

- for other options use the help function:

```
./waf --help
```

Build and Install

Just type:

```
./waf build
./waf install
```

or just

```
./waf build install
```

2.3.4 Using an IDE (Integrating Development Environment)

Different IDE's can be used with the waf build system. I prefer the platform independent Codelite IDE (`http://codelite.org/`).

- to get a Codelite project just type (after the configuration step):

  ```
  ./waf codelite
  ```

- open the codelite ide and open the codelite workspace file in the **dev** path
- **compile and install**: click with the *right mouse button* to your project (e.g. **module example**) in your Workspace View: **Custom Build Targets –> Build and Install**.
- debugging works if your configuration allows that

Chapter 3

Model Configuration

3.1 Quick Console Tutorial

3.1.1 Creating a test project

- open a python console (type **python** in the terminal or create a **python script** in your favorite editor):

```
import os,xpn_cfg_manager

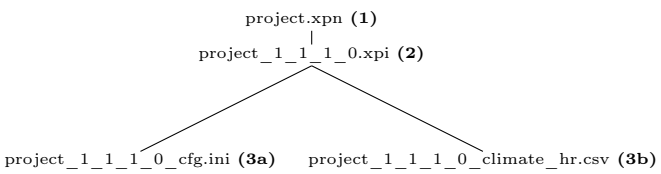
# setup some path variables for Expert-N:
xpn_path = os.environ['XPN_PATH']
base_path = xpn_path+"/built"
template_path = base_path+"/cfg_template"

# Name and Path of the new Expert-N Project:
project_name = base_path+"/cfg/test_project/test_project.xpn"

# create an empty project:
p=xpn_cfg_manager.cxpn_project(base_path=base_path,project_filename=
    project_name)
p.create_new_project(project_name,template_path+"/project.xpn")

# let's create add new grid with one Expert-N Column (Grid_Act,k,i,j =
    1,1,1,1):
col_name = p.add_new_grid(template_path+"/project.xpn",template_path+"/
    project.xpi","",1,1,1,1)
```

- tree structure of the project (project file with all dependences):



- The files of the project consist of:

1. the main project file (ini file): contains used plugins, Expert-N columns, global project properties
2. the config files for the Expert-N column (g,i,j,k=1,1,1,0): contains model choice of the specific column, model properties (model properties can point to additional config files)
3. additional config files for the chosen models: In this case: a) location and soil properties (ini), b) climate data (csv)

3.1.2 Run the test project

- running a test project with OpenMP (terminal):

```
expertn_bin --base-dir=$XPN_PATH/built --config-file=$XPN_PATH/built/cfg/
   test_project/test_project.xpn
```

- running a test project with MPI using 1 processors (terminal):

```
mpirun -n 1 expertn_bin --base-dir=$XPN_PATH/built --config-file=$XPN_PATH/
   built/cfg/test_project/test_project.xpn
```

3.1.3 Add a plugin (libhydrus.so), activate the hydrus flow module and set the boundary condition

- The following python script shows how to add a plugin and choose one model of this plugin:

```
# add a module plugin (*.dll,*.so) to the xpn_project (hydrus):
config = xpn_cfg_manager.ConfigParser()
config.read(project_name)
oldlibs = config.get("modul","libs")
config.set("modul","libs",oldlibs+"hydrus;")
configfile = open(project_name, 'wb')
config.write(configfile)
configfile.close()

#activate hydrus flow module(to 1,1,1):
config = xpn_cfg_manager.ConfigParser()
config.read(col_name)
try:
  config.add_section("flow module")
except:
  pass
config.set("water","flow module","HYDRUS Flow")

#set up options --> for standard values open the files in the template folder
    or pdf in doc folder
try:
  config.add_section("hydrus");
except:
  pass
```

```
config.set("hydrus","bottombc",1)
config.set("hydrus","mobil",0)
config.set("hydrus","infiltration_limit",1)
config.set("hydrus","infiltration_layer_limit",0)
configfile = open(col_name, 'wb')
config.write(configfile)
configfile.close()
```

- How do I get to know the models which are contained in a plugin (**module.so** or **module.dll**)? Just type in the terminal (using the *–show-modules* option):

```
expertn_bin --show-modules --base-dir=$XPN_PATH/built --config-file=$XPN_PATH
   /built/cfg/test_project/test_project.xpn
```

This will give you the following result:

group	sub group	plugin name	possible models
control	database	libexpertn_database.so	Expert N Standard Database Expert N Standard Read INI
water	hydraulic functions	libhydraulic_functions.so	Hutson and Cass van Genuchten and Mualem Brooks and Corey Brutsaert and Gardner Biomodal van Genuchten
control	pedotransfer	libpedotransfer.so	Campbell Vereecken Brakensiek & Rawls
control	output	xpn_output.so	Campbell XPN_OUTPUT
water	flow module	libhydrus.so	HYDRUS Flow

- How do I know the options of one model (e.a. **libhydrus.so**)?
  - if there are options, you will find the possible parameter in the template path. For **libhydrus.so**, open the file **$XPN_PATH/built/cfg_template/libhydrus.cfg**. It can contain options for one model (**HYDRUS Flow**), options for the complete model group in the plugin (**hydrus**) and options, which can be set as global options (**global**). In this case the parameters are chosen for the whole models in the plugin (libhydrus.so has only one model):

```
[hydrus]
bottombc = 1
mobil = 0
infiltration_limit = 1
infiltration_layer_limit = 1
```

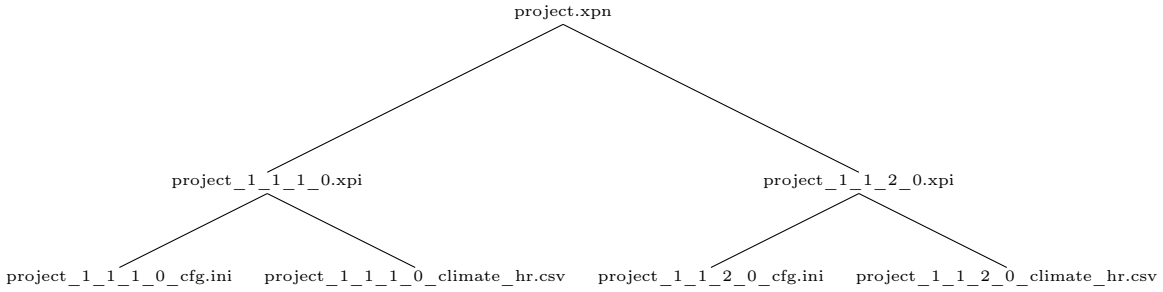
  - Explanation of the model options can be found in the documentation of the plugin, which should be found in: **$XPN_PATH/built/doc/libhydrus.pdf**.

3.1.4 Copy an Expert-N column (with all dependences and properties) and change the saturated conductivity of the soil in this Expertn-N column

- Copy a column:

```
# create a second Expert-N column (copy the old one, including all ini/cfg files)
    :
regstr = xpn_cfg_manager.get_regstr_from_filename(col_name)
col_name2 = p.add_new_grid(project_name,col_name,regstr,1,1,1,2)
```

- Now the dependency tree looks like that:



- Change soil properties:

```
#change the soil properties (saturated conductivity [mm/d]) of the second Expert-
    N column (1,1,2):
regstr = xpn_cfg_manager.get_regstr_from_filename(col_name2)
config = xpn_cfg_manager.ConfigParser()
config.read(col_name2)
ini_name_frame = config.get("Expert␣N␣Standard␣Read␣INI","filename")
ini_name = xpn_cfg_manager.replace_std_template_and_path(ini_name_frame,
    project_name,base_path,regstr)
config.read(ini_name)
cond_sat=[3811.6,3515.4,952.3,5925.6]
config.set("soil","cond_sat",';'.join(map(str, cond_sat))+';')
configfile = open(ini_name, 'wb')
config.write(configfile)
configfile.close()
```

3.2 Quick GUI Tutorial

- The OpenMP Version of Expert-N can also be controlled by the graphical interface, to open the GUI type:

```
expertn_gui
```

Chapter 4

Driver and Plugin Structure

4.1 Driver Implementation

4.2 Plugin Structure and Example

Bibliography

R. Stenger, E. Priesack, G. Barkle, G. Sperr, A tool for simulating nitrogen and carbon dynamics in the soil-plant atmosphere system, in: Proceedings of the technical session, 20, 19–28, 1999.

W. Skamarock, J. Klemp, J. Dudhia, D. Gill, D. Barker, Coauthors, 2008: A description of the Advanced Research WRF version 3. NCAR Tech, Tech. Rep., Note NCAR/TN-4751STR, 2008.

J. Hutson, R. Wagenet, LEACHM, Leaching estimation and chemistry model, version 3, Dep. of Soil, Crop and Atmospheric Sci. Res. Ser (92-3).

J. Simuunek, K. Huang, M. T. Van Genuchten, The HYDRUS code for simulating the one-dimensional movement of water, heat, and multiple solutes in variably-saturated media. Version 6.0, Res. Rep 144 (1998) 142.

T. Schaaf, E. Priesack, T. Engel, Comparing field data from north Germany with simulations of the nitrogen model N-SIM, Ecological Modelling 81 (1) (1995) 223–232.

R. Horton, S.-O. Chung, J. Hanks, J. Ritchie, et al., Soil heat flow., Modeling plant and soil systems. (1991) 397–438.

B. Nicolardot, J. Molina, C and N fluxes between pools of soil organic matter: model calibration with long-term field experimental data, Soil biology and biochemistry 26 (2) (1994) 245–251.

J. H. Thornley, et al., Grassland dynamics: an ecosystem simulation model., CAB international, 1998.

F. Chen, J. Dudhia, Coupling an advanced land surface-hydrology model with the Penn State-NCAR MM5 modeling system. Part I: Model implementation and sensitivity, Monthly Weather Review 129 (4) (2001) 569–585.

W. J. Parton, M. Hartman, D. Ojima, D. Schimel, DAYCENT and its land surface submodel: description and testing, Global and planetary Change 19 (1) (1998) 35–48.

S. Hansen, H. Jensen, N. Nielsen, H. Svendsen, Simulation of nitrogen dynamics and biomass production in winter wheat using the Danish simulation model DAISY, Fertilizer Research 27 (2-3) (1991) 245–259.