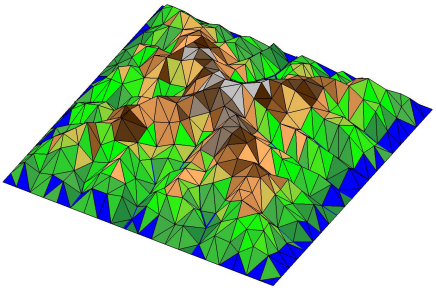

SIGNUM

Simple Integrated Geomorphological NUmerical Model

Version 1.0

User Manual

A. Refice

(refice@ba.issia.cnr.it)

Istituto di Studi sui Sistemi Intelligenti per l'Automazione (ISSIA),
Consiglio Nazionale delle Ricerche (CNR)
Via Amendola 122/O, 70126 Bari, Italy

January 2012

Contents

1. Introduction

This document is a manual for the SIGNUM landscape evolution model. The acronym SIGNUM stands for Simple Integrated Geomorphological NUmerical Model, and is a TIN-based landscape evolution model written in Matlab code.

The software is distributed under the GNU GPL license version 3 (see appendix A). Its documentation (this manual) is distributed under the Creative Commons license (see appendix B).

A thorough description of the software operation, as well as its scientific basis and some example applications can be found in publications such as [3], or [2].

2. Downloading and installing SIGNUM

The SIGNUM code can be downloaded from the Surface Processes and Quantitative Geomorphology Group website.

Please always refer to the above site for latest software updates. We'd be also glad to receive news about bugs (and possibly fixes!), modifications, and any other use of the software.

The package comes in a zip file containing all the necessary .m files to run the code. For best results, unpack/unzip the file in a separate directory, then add the directory to your Matlab path (see the Matlab documentation for how to do this). In this way, you will be able to access the SIGNUM program files while working in any other directory, which is useful to handle various simulations and projects with possibly different parameters and processes.

3. Basic operations

If the SIGNUM installation directory is in your Matlab path, running SIGNUM is as simple as starting Matlab and typing `SIGNUM` at the Matlab command prompt. This starts the SIGNUM model with the default parameters, which generate a sample surface of size 1000 m $\times$ 1000 m, subject to uniform uplift, uniform linear diffusion and uniform stream-power erosion, with nominal parameters. These values are used only as a representative test case. Normal operation will almost invariably require setting up parameters in a custom parameter file to be passed to the program. To use custom parameters, it is possible to specify an alternative parameter file name, which must be an m-file in the current path or in the Matlab path. So, for instance, a command such as `SIGNUM('myparameters')` can be entered at the command prompt, which says to SIGNUM to use the file `myparameters.m` as parameter file.

Once started, the program starts producing output, which consists of:

- a series of strings on the Matlab command window;
- a series of .mat files in the current directory;
- (optionally) a figure window showing the evolving surface, updated periodically.

A typical series of output messages obtained by running the default parameters is shown in fig. 1. It consists of a string summarizing information about the current

```
>> SIGNUM
** No parameter file selected:             **
** SIGNUM starting with default parameters **
Starting main cycle on 1000000 years
SIGNUM -- elapsed time: 0.00 years
SIGNUM -- elapsed time: 860.78 yr, dt = 860.78 yr, dz_min = 0.829 m, dz_max = 0.863 m
SIGNUM -- elapsed time: 1627.06 yr, dt = 766.28 yr, dz_min = 0.458 m, dz_max = 0.768 m
SIGNUM -- elapsed time: 2255.85 yr, dt = 628.79 yr, dz_min = 0.333 m, dz_max = 0.630 m
SIGNUM -- elapsed time: 3022.13 yr, dt = 766.28 yr, dz_min = 0.301 m, dz_max = 0.767 m
......
......
......
SIGNUM -- elapsed time: 116985.75 yr, dt = 396.47 yr, dz_min = -0.004 m, dz_max = 0.004 m
SIGNUM -- elapsed time: 117382.22 yr, dt = 396.47 yr, dz_min = -0.004 m, dz_max = 0.004 m
SIGNUM -- elapsed time: 117778.69 yr, dt = 396.47 yr, dz_min = -0.004 m, dz_max = 0.004 m
SIGNUM -- elapsed time: 118175.16 yr, dt = 396.47 yr, dz_min = -0.004 m, dz_max = 0.004 m
(apparent) Steady State reached!
>>
```

Figure 1: Typical output message sequence of the default SIGNUM operation.

SIGNUM run, then a variable number of strings with information about the ongoing simulation, such as the current simulated time, the used time interval, the minimum and maximum surface variations. More information for each time step can optionally be printed out in verbose mode (see sect. 4). Such statements are printed for every simulation step until end of the simulation, which will occur after a variable number of steps, when either the total number of simulated years, or an (apparent) steady state condition are reached. The latter is given by the maximum absolute surface variation being below a predefined threshold (see below for details). A typical final surface evolved by using the default parameter file is shown on the front page of the present manual. This particular example represents a surface of size 1000 m $\times$ 1000 m, sampled randomly but uniformly with maximum point spacing of 50 m, evolved until steady state under a combination of diffusion, channeling and uplift processes. Note that the vertical scale is exaggerated of a factor of 100 with respect to the horizontal dimensions.

4. Parameter file

SIGNUM is designed to be controlled entirely through a parameter file, which consists of a standard Matlab .m file script containing a series of variable definitions. Note that any custom parameter file will be executed after definition of the default values for all input variables in the main program, so any variable not defined in the parameter file is assumed to be set at its default value. To avoid missing overriding statements, the user is advised to use a parameter file template, such as the one listed in fig. 2, which can be modified and customized at will.

The main parts of the template parameter file are described in the following.

4.1. GTL module calls

The first three statements define the process modules. The process module calls `DIFFUSION_MODULE_CALL`, `CHANNELING_MODULE_CALL`, and `UPLIFT_MODULE_CALL` are strings containing standard Matlab calls to functions implementing the three basic types of geomorphic transport laws (GTL). The process modules are defined in detail in sect. 5.

```
%% Call function: SIGNUM('SIGNUM_parameter_file_template');

%     Copyright (C) 2011-2012  the authors
%
%     This program is free software: you can redistribute it and/or modify
%     it under the terms of the GNU General Public License as published by
%     the Free Software Foundation, either version 3 of the License, or
%     (at your option) any later version.
%
%     This program is distributed in the hope that it will be useful,
%     but WITHOUT ANY WARRANTY; without even the implied warranty of
%     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%     GNU General Public License for more details.

%% ------------------------------------------------------- GTL Module calls
% Call lines for the GTL modules to be used by the model.
% Output names, and dt final input name are mandatory.
% Do not forget the final semicolon within the quotes
% to avoid spurious screen output!
DIFFUSION_MODULE_CALL = '[dz,dtnew] = SIGNUM_diffusion_linear(kd,dt);';
CHANNELING_MODULE_CALL = '[dz,dtnew] = SIGNUM_streampower(kc,mc,nc,dt);';
UPLIFT_MODULE_CALL = 'dz = SIGNUM_uplift(uf,dt);';

%% ------------------------------------------------- Processing parameters
Run_name = 'SIGNUM_test';
% Start_time = 300037; <- uncomment to start from an existing surface
if exist('Start_time','var')
    load([Run_name,'__Time_',num2str(Start_time),'.mat']);
elseif exist([Run_name,'_initial_surface.mat'],'file');
    disp(['Found file ',[Run_name,'_initial_surface.mat'],': loading']);
    load([Run_name,'_initial_surface.mat']);
end
thresh_river = 5e4; % threshold contrib. area for plotting blue rivers
plot_surface = 1000; % plot surface every ... years
output_mat_file = Run_name;
logfilename = [Run_name,'.log'];
N_save = 10000; % save surface every ... years
N_epochs = 1000000; % number of simulation years
verbose = 0;
dzSS = 0.00001; % m/year; <- Threshold max. height change for steady state

%% ---------------------------------------------------------- Surface setup
[x,y,z,b] = SIGNUM_create_surface(2,... % 1 perturbed grid, else uses PDE
    [0, 1000, 0, 1000], ... % coordlims,
    50, ... % points spacing,
    0.01, ... % perturb_xy (not valid for PDE method),
    0, ... % z_slope,
    1.0,... % z_noise,
    0); % bcode:
%         0 - all 4 boundaries open,
%         1 - left and right (x=0,x=max) boundaries open, the rest closed
%         2 - upper and lower (y=0,y=max) boundaries open, the rest closed
%         3 - all 4 boundaries closed, except (0,0) corner, which is open

%% ------------------------------------------------------- GTL parameters
% --- Linear diffusion parameters
kd = 1e-2; % (linear) diffusivity constant (m^2/year)

% --- Channeling parameters
kc = 1e-4; mc = 0.5; nc = 1;

% --- Uplift parameters
uf = 1e-3; % constant uplift rate (m/year);
```

Figure 2: SIGNUM parameter file template.

4.2. Processing parameters

In this section some miscellaneous parameters and states are defined.

Run_name. This string is used as an identification at the beginning of the names of the output files generated by the model.

Start_time and check for previous data. The next part contains the check for any existing surface file. First, existing SIGNUM output files with assigned time evolution number are checked for existence. The string `Start_time` is used to this end. In the listing in fig. 2 this statement is commented. Suppose you have a SIGNUM run which has been run until a simulated time of, say, 300 ky. Suppose the exact (integer) number of simulated years is 300037, as in the example. You want to evolve this surface for some further simulated time. You should then look for a SIGNUM output file with the form `<Run_name>__Time_300037.mat`. In this case, assign the correct `<Run_name>` variable, then the correct `Start_time` value. SIGNUM will then load the existing .mat file and start to evolve the surface from the existing configuration, with the parameters specified in the subsequent parts of the parameter file.

thresh_river. This is the threshold to be used in the figure plot to highlight rivers with blue lines. Mesh node links in the TIN originating from points with contributing areas greater than this threshold are plotted as rivers, i.e. as blue lines with thickness proportional to the contributing area. If this variable is set to zero, an automatic threshold is used by the plotting function.

plot_surface. The time frequency of update for the graphical representation of the simulated surface. Setting this to zero disables surface plotting.

output_mat_file. Name "kernel" string for the output .mat files. In the example, the variable is set equal to the `Run_name` string, to have consistent file names.

log_file_name. If set to a nonempty string, a log file name is generated through Matlab's `journal` function, containing transcription of all the command window contents in the session.

N_save. Frequency at which output .mat files are saved. Note that, due to the Courant limit, the actual value of the simulated times at which output file save will occur will fluctuate. For instance, is `N_save` is set to 1000 years, output files will have names such as 1003, 2028, etc.

N_epochs. The number of years the simulation is scheduled to last. The simulation will go on until `N_epochs` years have been simulated, unless steady state is reached earlier.

verbose. Flag to output more information strings on screen during execution. The messages are also recorded in the log file if the `log_file_name` variable is set.

dzSS. This is the maximum absolute surface height variation. Whenever maximum absolute surface height variation between two successive iterations falls below this value, the simulation is stopped and a message issued that an (apparent) steady state has been reached.

4.3. Surface setup

In this section the initial model surface is set up. In the template example, an ancillary function, `SIGNUM_create_surface`, is used (see sect. 8.1 for further details). This can be substituted by other procedures for setting up triangulated surface, e.g. by loading external DEM data.

4.4. GTL parameters

This section of the parameter file contains the relevant parameters for the three GTL functions listed at the beginning. All the input parameters for each GTL function must be defined here, except for the `dt` variable.

5. Process modules

Surface height changes at each iteration are computed by applying a series of geomorphic transport laws (GTL), which are coded as Matlab functions acting on the (global) `points` structure, as described below. Three basic types of GTL are foreseen in the present version of the code, namely diffusion, channeling and uplift. These are assigned through the `DIFFUSION_MODULE_CALL`, the `CHANNELING_MODULE_CALL`, and the `UPLIFT_MODULE_CALL` strings assigned in the parameter file, as shown in sect. 4. The default functions for each of these three process types are described in the following.

5.1. `SIGNUM_diffusion_linear`

This function calculates height variations according to the formula

$$\frac{\partial z}{\partial t} = k_d \nabla^2 z \tag{1}$$

(see eq. (3) in [3]). Surface curvature ($\nabla^2 z$) is approximated through computation of the surface slopes on the arcs L_{ij} of the (usually Delaunay) triangulation composing the TIN, the Voronoi polygon side lengths λ_{ij} and the point contributing area A_i, as:

$$\frac{\partial z_i}{\partial t} = \frac{k_d}{A_i} \sum_{j=1}^{n} \lambda_{ij} \frac{z_j - z_i}{L_{ij}}, \tag{2}$$

(see eq. (4) and references in [3] for justification and details). Here the contribution to height variation of the point i is calculated by summing contributions over all $j = 1, \ldots, n$ prime neighbors of point i. The k_d parameter should be assigned in the parameter file (see the corresponding section and the `kd` assignment in the template parameter file in fig. 2. The correct syntax to be used in the assignment of the `DIFFUSION_MODULE_CALL` string inthis case is, as shown, `'[dz,dtnew] = SIGNUM_diffusion_linear(kd,dt);'`. The `dt` input variable represents the current value of the time step interval; the outputs are the array `dz` of the height changes due to the diffusion GTL, and the time step `dtnew` to be used in the next time step. The latter is calculated within the function as a function

of the surface states and parameters, in order to satisfy the Courant condition[1] (see [3] for details). In the present version, the kd parameter can be defined as a spatial variable, in the sense that it can be a vector of the same length as the `points` structure, with one value for each surface point. In this way, spatially-variable diffusion constants can be simulated.

5.2. `SIGNUM_streampower`

This function calculates height variations according to the formula

$$\frac{\partial z}{\partial t} = -K_c A^m (\nabla z)^n . \tag{3}$$

(see eq. (9) and references in [3] for justification and details). Here again the surface slope is approximated over the edges of the (Delaunay) TIN mesh, and the contributing area A is calculated over the Voronoi polygons, as:

$$\frac{\partial z_i}{\partial t} = -K_c A_i^m \left(\frac{|z_i - z_j|}{L_{ij}} \right)^n , \tag{4}$$

(see eq. (10) in [3]). K_c, m and n are model parameters specified in the parameter file (see fig. 2). The correct syntax to be used in the assignment of the `CHANNELING_MODULE_CALL` string in this case is, as shown, `'[dz,dtnew] = SIGNUM_streampower(kc,mc,nc,dt);'`. The `dt` input variable represents the current value of the time step interval; the outputs are the array `dz` of the height changes due to the channeling GTL (in this case, the stream-power law), and the time step `dtnew` to be used in the next time step. The latter is calculated within the function as a function of the surface states and parameters, in order to satisfy the Courant condition (see [3] for details).

In the present version, any of the `kc`, `mc` and `nc` parameters can be defined as spatial variables, in the sense that they can be vectors of the same length as the `points` structure, with one value for each surface point. In this way, spatially-variable stream-power erosion conditions can be simulated.

5.3. `SIGNUM_uplift`

This function calculates height variations due to a global rule, i.e. a spatial field of constant uplift of all the points representing the surface:

$$\frac{\partial z}{\partial t} = u_f, \tag{5}$$

where u_f is a parameter to be assigned in the parameter file. The correct syntax to be used in the assignment of the `UPLIFT_MODULE_CALL` string in this case is, as shown, `'dz = SIGNUM_uplift(uf,dt,1);'`. The `dt` input variable represents the current value of the time step interval; the outputs are the array `dz` of the height changes due to the uplift GTL (in this case, constant uplift). In the present software version, the `uf`

[1] The Courant-Friedrichs-Lewy (CFL) condition, or Courant condition for short, is an upper limit condition on the simulation time step, which applies in general to the solution of partial differential equation problems through explicit finite-elements methods.

parameter can be defined as a spatial variable, in the sense that it can be a vector of the same length as the `points` structure, with one value for each surface point. In this way, spatially-variable uplift patterns can be simulated.

6. Developing new modules

The SIGNUM model is designed to be easy to understand, tweak, and customize. As in this initial version only basic modules and processes are implemented, we strongly encourage hacking and adaptation of process functions, implementation of innovative GTLs and algorithms, as well as any other improvement, from the scientific community.

To develop new GTL modules, first write a new .m Matlab function implementing the GTL as a function of the states stored in the `points` global structure (see below), as well as any necessary parameter. To perform this step, make sure that the calling syntax for the new GTL function is consistent with the default GTL functions described above, in particular that one of the input is the current time step `dt`, and one of the outputs is the GTL contribution to the height variation, `dz`. If transient effects are implemented, such as in the default cases of diffusion and channeling erosion, another output should be the new maximum time step allowed by the Courant condition, `dtnew`.

Then, substitute the new function call in your custom parameter file, e.g. modifying one of the three module calls, and add/modify all the necessary parameters in the corresponding section of the same file.

If the new function needs to define new fields in the `points` structure, these can be easily added or modified by editing the `SIGNUM_init_points` function.

Many other improvements can be conceived, based on this skeletal recipe, so feel free to experiment. We'd be happy to receive notice of any use of the model, so feel free to contact the author via e-mail.

6.1. The `points` structure

The `points` structure is the basic array containing all the states of the simulated surface at any time step. It is defined as a global variable, so that it is not necessary to pass it as an argument to the various sub-functions. This solution is often used in Matlab as a valid substitute for the standard pass-by-reference procedure available in other languages to avoid unnecessary memory storage. The `points` structure consists of a series of records, one for every surface point, with fields containing information related to that point. The fields present in the structure in this software version are reported in tab. 1, reproduced from [3]. The `points` structure is built by a specific .m function, called `SIGNUM_init_points`. This function can be inspected and edited/customised at will to add further fields. Note that mesh edges, as well as all edge-related quantities, such as slopes, etc., are stored point-wise, thus doubling the space requirements. Future releases will probably address this issue.

7. Output files

The main output of the model is a series of .mat files, each containing the entire Matlab memory content at a particular simulated time step. The files have names consisting

Table 1: Field names of the points structure. The comments describe the entry and its size. For instance, the field `points(`i`).slopes` is an array containing the values of the slopes of all the Delaunay edges connected to point i (after [3]).

Field name	Comment
`x,y,z`	coordinates [scalars]
`adjacents`	indices of all adjacent points, [1-D array of variable size n_i]
`slopes`	slope value for every edge [1-D array of variable size n_i]
`delaunay_edges`	lengths of planar Delaunay edges connected to each point [1-D array of variable size n_i]
`voronoi_edges`	lengths of Voronoi edges of each voronoi polygon [1-D array of variable size n_i]
`voronoi_area`	area of Voronoi polygon [scalar]
`voronoi_coordinates`	coordinates of Voronoi vertices [1-D array of variable size n_i]
`contrib_area`	contributing area of each point [scalar]
`contrib_points`	indices of upslope contributing points [1-D array of variable size]
`contrib_flux`	flux from contributing points [1-D array of variable size]
`z_following`	height of following point [scalar]
`following`	index of following point [scalar]
`preceding`	indices of preceding points [1-D array of variable size]
`border`	border flag (0,1 or 2) [scalar]

of a predefined string, which is set in the parameter file (default is `'SIGNUM_output'`), then the `'__Time_'` string (note the double underscore), and a final part consisting of the current evolution simulated time, rounded to the nearest integer.

An initial .mat file is also generated, with the "time" string substituted by the string `initial_surface`, containing the initial surface states, unless the model is run by loading an existing surface, e.g. if the `Start_time` is set in the parameter file (see sect. 4).

If the `logfilename` variable is set, also a text log file is output, containing all the screen output by the program.

8. Ancillary and visualization functions

Some ancillary functions are present in the SIGNUM distribution to help create and manage simulated TIN surfaces. They are described in the following.

8.1. `SIGNUM_create_surface`

This function creates three vector arrays representing the x, y, z coordinates of a synthetic surface, as well as a boundary flag vector. The points coordinates are sampled uniformly to form a TIN surface of rectangular shape, with user-controlled characteristics. Two modes of generation of the surface points can be used, one which makes use of the Matlab Partial Differential Equation (PDE) Toolbox function `initmesh`, which generates a mesh of points inside closed regions of any shape (see Matlab PDE Toolbox documentation for further details), and another which only uses default functions, and

creates the points by perturbing a regular, rectangular array of points by random quantities. The choice of the generation mode is controlled by the `code` flag. Additional inputs control the size of the surface and the maximum or mean spacing of points.

An initial uniform tilt can be assigned to the surface in the x direction through the `z_slope` parameter, or a wedge shape if `z_slope` is a 2-valued array.

A gaussian-distributed noise can be added to the surface elevation, with st.dev. controlled by the `z_noise` parameter.

Finally, a `bcode` flag controls the state of the four boundary sides of the surface, which affect the subsequent simulation: open boundaries are usually kept to low height values (e.g. zero) so that sediment and water are allowed to flow through them and out of the surface, closed boundaries are treated as delimiting periodic boundary conditions, so material and/or water flow is forbidden through them. Four configurations are foreseen for the sides of the rectangular surface, namely generating either all 4 sides open (`bcode` = 0), left and right sides open and the rest closed (`bcode` = 1), upper and lower sides open and the rest closed (`bcode` = 2), or all sides closed except for the single point in the origin ($x = y = 0$), which is kept open and serves as single surface outlet (`bcode` = 3).

The `SIGNUM_create_surface` is used typically within the parameter file to easily generate synthetic surfaces of various kinds.

Note that, in this software version, only the point coordinates and boundary codes have to be created within the parameter file and passed to the main program. The TIN structure (Delaunay triangulation and Voronoi polygon structure) is built within the SIGNUM program through the `SIGNUM_assign_structure` sub-function, which is essentially a wrapper to the Matlab `delaunay` triangulation function.

8.2. `SIGNUM_draw_TIN`

This function draws any TIN simulated through SIGNUM as a 3-D surface, using a "topographic" colormap, and optionally highlights rivers as TIN edges connecting points with cumulated drainage area above a given threshold.

The function accepts as input the SIGNUM `points` structure as defined in the main program (see sect. 6.1), as well as an optional threshold for river plotting. The twin function `SIGNUM_draw_TIN_g` requires the presence of the global `points` structure in memory. Typical usage is by first loading one of the output .mat files relative to a given simulated time step, then launching the function, with only one optional argument – the threshold area.

8.3. `SIGNUM_cascade`

This function implements the "cascade" algorithm described in [1]: it is used within the SIGNUM model to derive a correct visiting order of the TIN nodes to calculate the contributing areas. It has an optional second output which is an index array containing integer flags for detected hydrological basins: all points belonging to the same basin will share the same index in the `ncat` output array. This is useful to perform hydrologic / geomorphologic analyses on the simulated surfaces.

The function accepts as input the SIGNUM `points` structure as defined in the main program (see sect. 6.1). The twin function `SIGNUM_cascade_g` requires the presence of

the global `points` structure in memory.

8.4. `SIGNUM_plot_SA`

This function plots the slope vs. area log-log chart of all the points contained in a SIGNUM `points` structure. The twin function `SIGNUM_plot_SA_g` requires the presence of the global `points` structure in memory.

8.5. `SIGNUM_extract_river`

This function extracts the sequence of surface points from a given one, which is passed to the function as input, to a final point, which usually coincides with a surface outlet. The function requires the presence of the global `points` structure in memory. The input is an index pointing to one record of the `points` structure. The output is the sequence of indices of points hydrologically linked on the surface starting from the input point, i.e. all the points subsequently linked by the "following" attribute.

Point indices can be evidenced on a Matlab figure showing the TIN structure by using the callback function described in the next section.

8.6. `Callback_SelTINPointNumber`

This is a callback function which can be used as a substitute for the standard one in the Matlab function showing the current SIGNUM mesh surface, to highlight a particular point index in the `points` structure. This is useful to identify a particular point on the surface., e.g. with the purpose of extracting all points downslope, through the `SIGNUM_extract_river` function described above in sect. 8.5.

To use it, make sure you have created a Matlab figure showing a SIGNUM surface as a TIN structure, e.g. through use of the above-described `SIGNUM_draw_TIN` function (see sect. 8.2). Then, click on the toolbar button "Data cursor". In the context menu, click on "Select Text Update Function...", then browse and select the `Callback_SelTINPointNumber.m` function within the standard "open file" dialogue and click OK. Now, clicking on a point on the TIN surface, you should see a pop-up window containing as additional information the current point index, indicated as "Pid".

References

[1] J. Braun and M. Sambridge. Modelling landscape evolution on geological time scales: a new method based on irregular spatial discretization. *Basin Research*, 9:27–52, 1997.

[2] D. Capolongo, E. Giachetta, and A. Refice. Numerical framework for geomorphological experiments. *Geografia Fisica e Dinamica Quaternaria*, 34:75–80, 2011.

[3] A. Refice, E. Giachetta, and D. Capolongo. SIGNUM: a Matlab, TIN-based Landscape Evolution Model. *Computers & Geosciences*, 2012. In press.

A. GNU GENERAL PUBLIC LICENSE

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. 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 them 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 prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. 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.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for

a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey 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; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users

beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

   b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

   c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

   d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

   e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

   A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

   A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be

resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added

by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright

holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If 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 convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU 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 that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    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.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS 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.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
```

```
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.

B. Creative Commons license