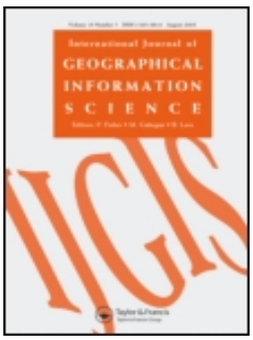


International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Dynamic environmental modelling in GIS: 2. Modelling error propagation

D. Karssenberg Corresponding author & K. De Jong

To cite this article: D. Karssenberg Corresponding author & K. De Jong (2005) Dynamic environmental modelling in GIS: 2. Modelling error propagation, International Journal of Geographical Information Science, 19:6, 623-637, DOI: 10.1080/13658810500104799

To link to this article: https://doi.org/10.1080/13658810500104799

International Journal of Geographical Information Science
Vol. 19, No. 6, July 2005, 623–637

Taylor & Francis
Taylor & Francis Group

Research Article

Dynamic environmental modelling in GIS: 2. Modelling error propagation

D. KARSSENBERG* and K. DE JONG

Department of Physical Geography, Faculty of Geosciences, Utrecht University, PO Box 80115, 3508 TC Utrecht, The Netherlands

(*Received 18 February 2003; in final form 20 August 2004*)

Environmental modelling languages provide the possibility to construct models in two or three spatial dimensions. These models can be static models, without a time component, or dynamic models. Dynamic models are simulations run forward in time, where the state of the model at time t is defined as a function of its state in a period or time step preceding t. Since inputs and parameters of environmental models are associated with errors, environmental modelling languages need to provide techniques to calculate how these errors propagate to the output(s) of the model. Since these techniques are not yet available, the paper describes concepts for extending an environmental-modelling language with functionality for error-propagation modelling. The approach models errors in inputs and parameters as stochastic variables, while the error in the model outputs is approximated with a Monte Carlo simulation. A modelling language is proposed which combines standard functions in a structured script (program) for building environmental models, and calculation of error propagation in these models. A prototype implementation of the language is used in three example models to illustrate the concepts.

Keywords: Dynamic environmental modelling; Error propagation; Spatio-temporal modelling; Stochastic modelling; Sensitivity analysis

1. Introduction

The first paper (Karssenberg and de Jong 2005) of this series of two papers showed that it is possible to extend existing two-dimensional dynamic environmental modelling languages with an additional spatial dimension, resulting in a tool for dynamic model construction in two and three spatial dimensions. The language described in the first paper is restricted to the case when inputs, parameters, and the model structure are exactly known. In many situations of modelling, this is not the case, because of, for instance, measurement errors, or insufficient field data. Also, it may be difficult to identify the model structure, since many different model structures can be used to represent a process. For these cases, it is said that the inputs, parameters, and the model structure are associated with uncertainty or errors, and it is important to know the effect of this uncertainty or errors on the output of the model.

Error-propagation modelling, also referred to as uncertainty analysis, allows the researcher to assess the error in the model output variables resulting from

*Corresponding author. d.karssenberg@geo.uu.nl

International Journal of Geographical Information Science
ISSN 1365-8816 print/ISSN 1362-3087 online
http://www.tandf.co.uk/journals
DOI: 10.1080/13658810500104799

propagation of model input errors through the model, or resulting from uncertainty in the model structure with its associated parameters (Heuvelink 1998, Crosetto and Tarantola 2001). It is an important issue in environmental modelling, since input data of environmental models have many different sources, with a wide range of errors associated with them (Thapa and Bossler 1992). The propagation of these errors through complicated dynamic spatial environmental models can only be assessed with powerful computational techniques. Most of these techniques model errors as stochastic variables (Heuvelink 1998). Analytical solutions for error propagation as a result of spatial functions on stochastic variables exist, but these are only available for a limited number of relatively simple functions and do not support most dynamic models. As a result, error propagation in dynamic spatial environmental models is mostly calculated by Monte Carlo simulation modelling (Heuvelink 1998). Although error-propagation modelling has long been a subject of concern in the GIS research community (reviewed by Heuvelink and Burrough 2002), a framework to implement it, let alone the implementation itself, in a dynamic spatial environmental-modelling language as described in the first paper (Karssenberg and de Jong 2005), is not yet provided. This lack of off-the-shelf tools for error-propagation modelling is somewhat strange, since the theory of spatial stochastic simulation and Monte Carlo simulation is widely described and generally accepted. What are missing are concepts and tools that make error-propagation modelling possible for a wide range of users. Including a standard Monte Carlo simulation modelling tool in GIS embedded dynamic spatial environmental modelling languages would make it easier for researchers to analyse the errors associated with model outputs, as well as performing sensitivity analysis (Crosetto and Tarantola 2001).

This paper deals with the development of a dynamic spatial environmental-modelling language that uses Monte Carlo simulation for error-propagation modelling. We do not aim here at generating theory on error-propagation modelling, since this is done elsewhere. Instead, we focus on language concepts of a language that encapsulates existing theory. First, the main concepts of error-propagation modelling and Monte Carlo simulation are described. Second, a description is given of the extensions which need to be made to the language described in the first paper (Karssenberg and de Jong 2005), in order to model error propagation. These extensions involve the entities and functions of the language, and its syntax. Finally, three example models built with a prototype of the language are given to illustrate how it can be applied, followed by a discussion.

2. Error-propagation modelling in spatial environmental models

Although the emphasis here is on dynamic spatial environmental models, having a time component, error-propagation modelling is as important for static spatial environmental models. A static spatial environmental model does not have a time component. It can be described by:

$$Z_{1..m} = \mathrm{f}(I_{1..n}, P_{1..l}) \tag{1}$$

with, $I_{1..n}$, the inputs, $Z_{1..m}$, the model variables, and a model structure defined by a function or set of functions f, with associated parameters $P_{1..l}$. Note that $I_{1..n}$, $Z_{1..m}$ and $P_{1..l}$, are defined in two or three spatial dimensions. An example of a static model is the derivation of infiltration capacity from a soil type map, or the calculation of distance from a vegetation island. As noted in the first paper

(Karssenberg and de Jong 2005), a dynamic spatial model is described in an analogous way:

$$Z_{1..m}(t+1) = \mathrm{f}(Z_{1..m}(t), I_{1..n}(t), P_{1..l}) \quad (2)$$

with the same meaning of the symbols as in equation (1), and t, the time, while the inputs and model variables have a value for each time step, here.

Both for static and dynamic spatial environmental models, model output error is caused by *input error* and *model error*. The input error is associated with uncertainty in model input variables ($I_{1..n}$ in equation (1) or (2)) for a given computational model, which are unique for a specific study site or period, such as elevation data, conductivity values or temperature time series data. The model error is associated with the discrepancy between the computational model, defined by f and $P_{1..l}$ in equations (1) or (2), being an approximation of reality. Model error refers to an incorrect choice of model equations and parameters for the representation of real-world processes. Since the analysis of model error is still in development, we focus on the propagation of input error, although the theory and prototype software proposed here can be used for the analysis of errors in the model parameters, too.

Most studies model error by representing model inputs, parameters, and variables as stochastic (random) fields in two or three dimensions. By doing so, the inputs and parameters of a model ($I_{1..m}$ and $P_{1..l}$ in equation (1) or (2)) become stochastic, and also most of the model variables $Z_{1..n}$ (equation (1) or (2)), because these are derived from the inputs. This kind of model, having stochastic model variables as a result of stochastic inputs and/or parameters, is referred to as a stochastic model. The aim of error-propagation modelling is to derive the probability distributions (or parameters describing these) of the stochastic model variables $Z_{1..n}(t)$ from the stochastic inputs $I_{1..m}(t)$ and parameters $P_{1..l}$, and to store the probability distributions of these model variables which are of interest, i.e. the model output variables. For complex environmental models, this can be approximated only with Monte Carlo simulation modelling (Heuvelink 1998). Monte Carlo simulation involves two steps (Hammersley and Handscomb 1979, Heuvelink 1998):

Step 1. For each Monte Carlo run s, $s=1..k$ (below, lower-case letters represent realizations of random variables given as upper-case letters in equation (1) or (2)):

a. generate realizations $p_{1..l}$ of each stochastic model parameter, and realizations $i_{1..m}$ for each stochastic input variable, for each time step t (in the case of a dynamic model).
b. with these realizations, run f(·) (equation (1) or (2)), and store the realizations of the model output variables $z_{1..n}$ in which the interest lies, in the case of a dynamic model for all time steps.

Step 2. Compute sample statistics (e.g. mean, variance, skewness) from the k model outcomes, for each model variable $1..n$ and each time step t, or for a selection of model variables and time steps.

This approach needs a stochastic description of model input variables with their associated errors, and a methodology to draw realizations from these stochastic model inputs, needed in step 1a of the Monte Carlo procedure. A review of the different types of error associated with the input data stored in GIS is given by NCDCDS (1988), Lanter and Veregin (1992) and Thapa and Bossler (1992). For most types of error, stochastic error models and associated numerical techniques for

drawing realizations are available. Attribute error for a continuous variable without spatial correlation (assuming a constant attribute value over an area) can be simulated as a random variable with a specified probability density function using standard algorithms (e.g. Press *et al.* 1986), while geostatistical software tools (e.g. Deutsch and Journel 1998, Pebesma and Wesseling 1998) provide algorithms for generating spatially correlated random fields, where correlation between variables can be accounted for, too. The same tools for geostatistics can be used to generate spatially correlated fields for categorical (classified) attributes, using indicator simulation (Bierkens and Burrough 1993). Other approaches to model errors in categorical attributes as stochastic variables are provided by Fisher (1991), Goodchild *et al.* (1992) and Veregin (1994, 1995). These are mostly based on the classification error matrix (Lanter and Veregin 1992). Positional accuracy is mainly important for environmental modelling as a source of class boundary uncertainty, and several stochastic methods have been described to deal with this (Davis and Keller 1997, Kiiveri 1997, Leung and Yan 1998). Another important type of error in environmental modelling is related to uncertainty caused by interpolation of time series data, which seems to be underexposed in research, although it is expected that techniques similar to these used for spatial data apply.

3. Extensions to the entities and functions of the language

3.1 *Entities*

As noted in the first paper (Karssenberg and de Jong 2005), two entities are used in the language: two-dimensional maps, with a spatial discretization in cells, and three-dimensional blocks, with a spatial discretization in voxels. Each cell on a map contains the same map variables, while each voxel in a block contains the same block variables. For deterministic dynamic modelling, the first paper proposes a fixed discretization of time in time steps, with a fixed length of time steps, for all variables. This is represented by a one-dimensional array of time steps in which each field contains a value of a variable for a certain map or block. For dealing with the Monte Carlo simulation, the same approach is followed, by adding an additional dimension to the array (figure 1). At the end of a model run, each field (t, s) in this array contains an attribute value for time step t and Monte Carlo sample s, for the cell at (x,y), or the voxel $V(x,y,v)$. Note that, in the case of a static model, the two-dimensional array with values shown in figure 1 reduces to a one-dimensional array, with Monte Carlo samples only.

3.2 *Functions of the language*

The first paper (Karssenberg and de Jong 2005) described functions of the language, which can be used to represent the function f in equation (1) or (2). For Monte Carlo simulation, stochastic functions and functions calculating descriptive statistics are needed in addition to these functions:

1. *Stochastic functions*. The stochastic functions assign a realization of a random variable to a map or block variable (for step 1a in the Monte Carlo simulation procedure). Different types of model input error are represented by different types of random variables with their associated realizations, as noted in section 2.

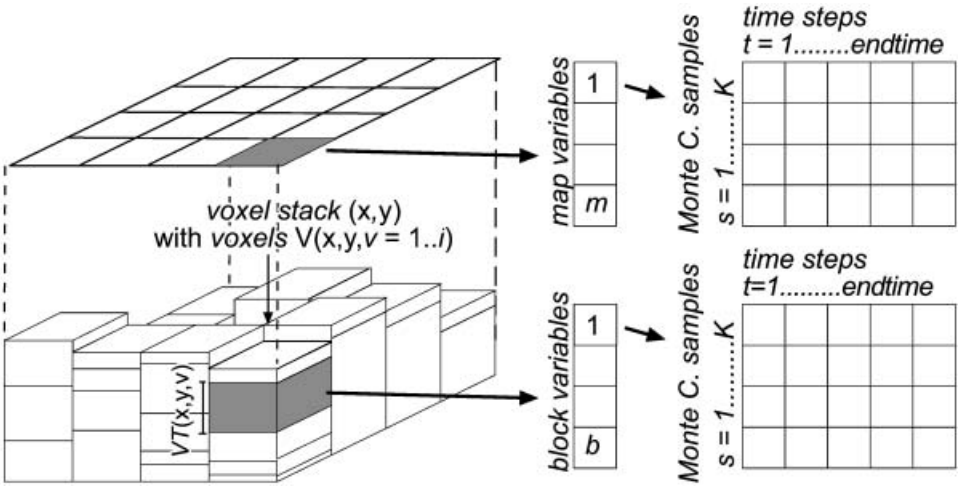


Figure 1. Entities of the language. Left: map and block; centre, list of map or block variables; right: matrix with values for each time step and each Monte Carlo sample, stored for each map or block variable, for each cell or voxel.

2. *Functions calculating descriptive statistics.* These functions calculate descriptive statistics of values within a certain group of values in the matrices attached to each cell or voxel (figure 1). Statistics to be calculated include, for instance average, quartiles or variance (Pebesma *et al.* 2001). As shown in figure 2, these statistics can be calculated as an aggregated value (1) over two- or three-dimensional space, such as average pollutant concentration of a lithological unit, per time step and Monte Carlo sample; (2) over time, such as the maximum pollutant concentration during a model run, per cell/voxel and per Monte Carlo loop; or (3) over Monte Carlo samples, such as the variance in pollutant concentration per cell/voxel and time step.

4. Syntax

4.1 *Functions*

The syntax of the stochastic functions and the functions calculating descriptive statistics is similar to the syntax of the other functions provided by the language (see Karssenberg and de Jong 2005). The stochastic functions assign a realization of a two- or three-dimensional random field to a map or block variable, respectively. Their input arguments define the statistical properties of the random field to draw the realizations from. For instance, a realization of a random field with spatial autocorrelation is defined by:

$$Result = \mathbf{spatcor}(Variogram, Input_{1...n}, Options_{1...n})$$

with *Result*, the output map or block variable containing a realization of a random field, *Variogram*, a code defining the variogram type and parameters of the random field, $Input_{1...n}$, conditioning data, and $Options_{1...n}$, options (e.g. as applied in Gstat, Pebesma and Wesseling 1998). For other types of random fields, similar notations can be developed.

The functions calculating descriptive statistics aggregate over space, time or the Monte Carlo dimension (figure 2) by calculating statistics such as average, standard

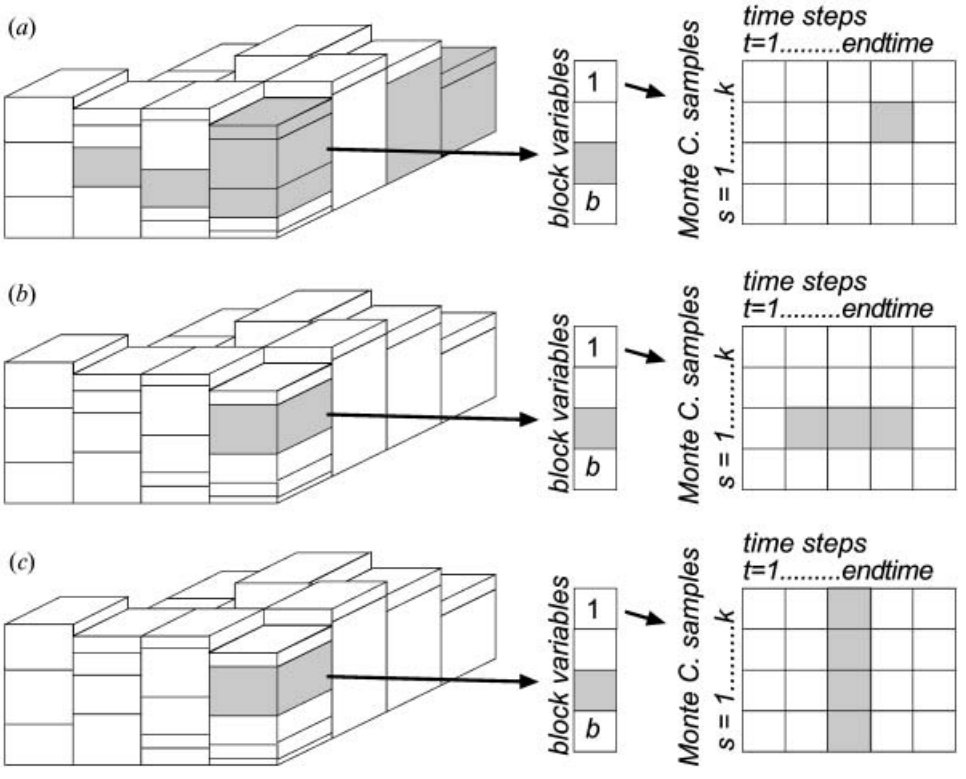


Figure 2. Functions calculating descriptive statistics, aggregating over (*a*) space, (*b*) time and (*c*) the stochastic dimension, i.e. Monte Carlo samples.

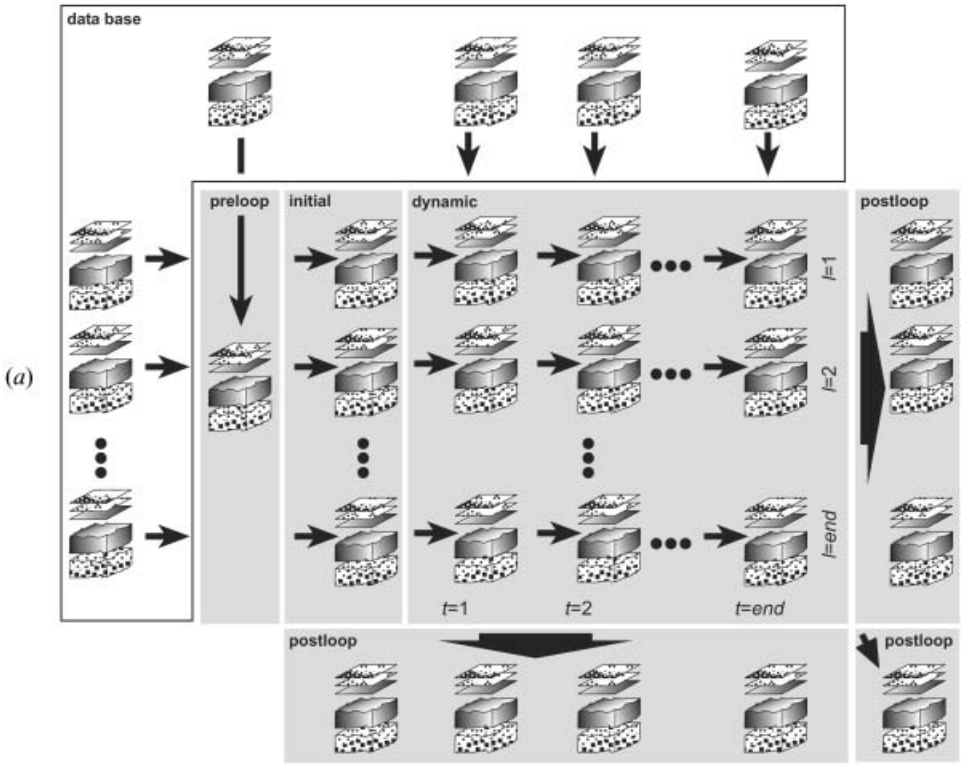
deviation, quartiles, for a map or block variable. The statistical functions aggregating over space calculate statistics of cell or voxel values belonging to the same spatial class (figure 2(*a*)), for each time step and each Monte Carlo sample. These functions use two inputs: the map or block variable for which the statistical value is calculated, and a map or block variable with classes defining the areas over which statistics need to be calculated. For instance,

```
AveZincLith = spaceaverage(Zinc, Lith)
```

calculates the average zinc concentration (block variable `Zinc`) over each lithological unit (block variable `Lith`), and assigns these average values to the block variable `AveZincLith`. This is done for each time step and each Monte Carlo sample. Using the same syntax, **spacesd** would calculate the standard deviation of the `Zinc` values per lithological unit. For aggregating over time, functions are provided calculating statistics over periods in time (figure 2(*b*)). For instance,

```
AveZincMonth = timeaverage(Zinc, Month)
```

calculates the average zinc concentration over periods defined by Month, which is a block variable containing different identifiers for each month. This monthly average is calculated for each voxel and each Monte Carlo sample. The same operation could be done on map variables. Aggregating over the Monte Carlo dimension (figure 2(*c*)) is done in a similar way, by calculating statistics over the k model outcomes (step 2 in the Monte Carlo simulation procedure), for each time step and



(b)

```
preloop
  # sequence of spatial functions
  ...

  initial (nrloops = ...)
    # sequence of spatial functions
    ...

    dynamic (nrtimesteps = ...)
      # sequence of spatial functions
      ...

postloop
  # sequence of spatial functions
  ...
```

Figure 3. (*a*) Concepts and (*b*) modelling script for temporal, two- or three-dimensional, and error propagation modelling.

each cell or voxel. For instance,

```
ZincQuartile = sampleperc(Zinc, 25)
```

calculates the first quartile (defined by the second argument `25`) of the zinc concentration Zinc, for each time step and voxel, based on all Monte Carlo samples.

4.2 *Script structure*

The script structure described in the first paper (Karssenberg and de Jong 2005) is extended to provide an additional loop, generating the k Monte Carlo samples, representing step 1 in the Monte Carlo simulation procedure. Figure 3 gives the

concepts of the extended script, structured in sections. Each section contains a list of operations, which are sequentially executed. The operations in the **preloop** section are executed only once, at the start of the model run. These read data from the database, generating map or block variables that are the same for each Monte Carlo loop. The **initial** and **dynamic** sections represent the temporal behaviour of the model. The **initial** section defines the initial state of the model at time step 0, reading data from the database or results of the **preloop** section. The **dynamic** section, representing the temporal change, is a section with functions that are run for each time step. The operations in the dynamic section represent the change in the state of model variables over one time step (equation (2)). For error-propagation modelling, the **initial** section and the **dynamic** section (for all time steps) are run for each Monte Carlo simulation loop. The statistical operations in the **postloop** section aggregate over the spatial, the time, and/or the stochastic (Monte Carlo simulation) domain. Note that a script does not always need to contain all these sections. For instance, in the case of a static model, the dynamic section is not used, as will be shown by the first example model.

5. Example models

5.1 *Two-dimensional spatial model*

The prototype implementation of the language described in the first paper (Karssenberg and de Jong 2005) is extended according to the concepts described here. Table 1 shows a script for a model simulating clonal growth of vegation, made with the prototype implementation of the language. Figure 4 gives the most important variables used in the script. A plant species occupies the area given as Boolean TRUE on the map variable `Ini` (figure 4(*a*)), read from the database in the preloop section. By clonal growth, the plant spreads over the whole area, resulting in a larger area occupied by the plant after 50 years, which is calculated in the initial section. The first operation in the initial section creates a map containing for each cell the time (years) needed for the plant to spread 1 m. The value and the error associated with this input parameter depend on the soil type. From a field

Table 1. Script for modelling clonal growth of vegetation, with remarks behind #.

```
preloop
  Ini='distr.map'                  # initial distribution of plant

initial (nrloops=500)
  # spreading time for peat (years)
  PeatYears=0.1+mapnormal()*0.001
  # spreading time for other soil types (years)
  OtherYears=0.5+mapnormal()*0.02
  # number of years needed to move the vegetation front 1 m
  Years=if(Peat then PeatYears else OtherYears)
  # time to colonization (yr)
  ColTime=spread(Ini,Years)
  # colonized after 50 years
  Col=ColTime<50

postloop
  # probability of colonization after 50 years
  ColProb=sampleprob(Col)
```

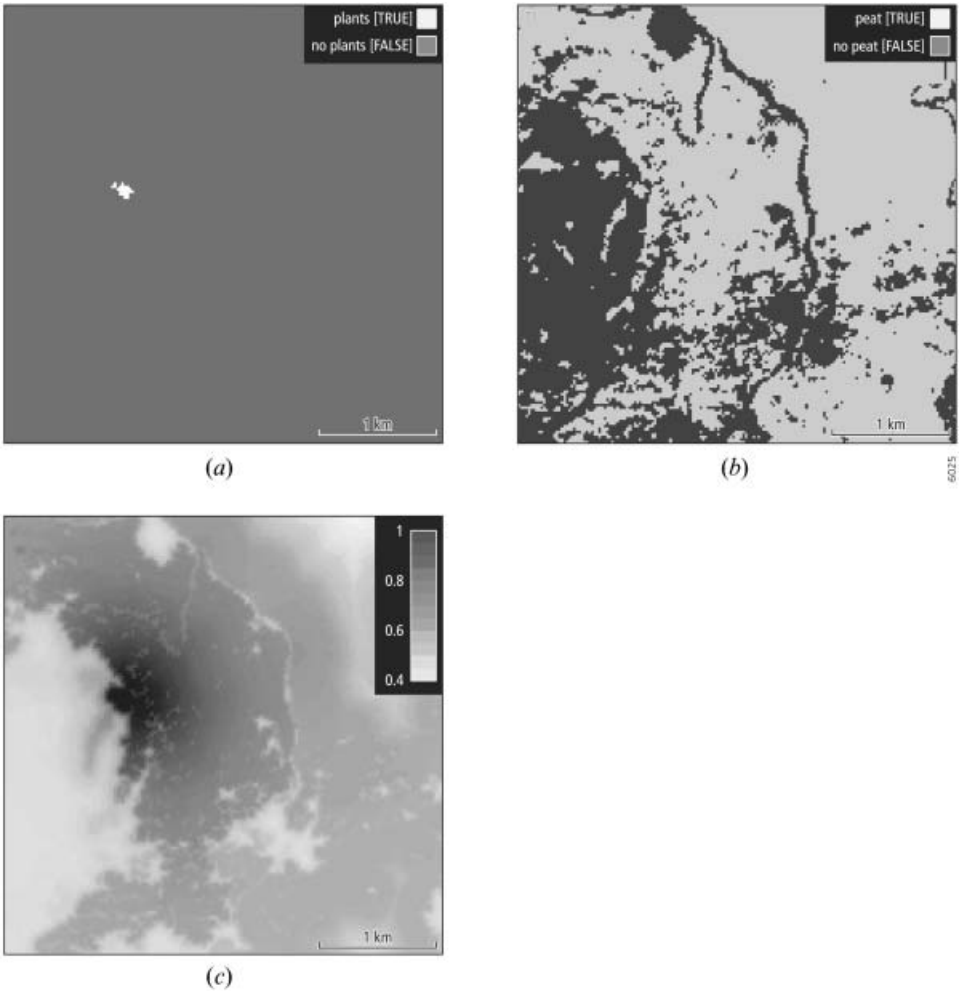


Figure 4. Clonal growth model. (*a*) Initial distribution of plant, variable name `Ini` in table 1, Boolean 1 (TRUE) represents growth location, 0 (FALSE) other locations. (*b*) Distribution of peat in the area (`Peat`), Boolean 1 (TRUE) represents peat, 0 (FALSE) other soil types. (*c*) Probability of colonization after 50 years, `ColProb`.

investigation, the value of the spreading time could be estimated as 0.1 year (standard deviation 0.001) for peat, and 0.5 (standard deviation 0.02) for other soil types. The two lines using the **mapnormal** function, drawing a value from a normal distribution with zero mean and standard deviation one, generate for each Monte Carlo loop a realization of a spreading time map with these statistical properties, independently for peat and for the other soil types, see table 1. These realizations are combined in one map with an **if..then** function, assigning `PeatYears` to the peat areas having a TRUE on the map variable `Peat` (figure 4(*b*)), and `OtherYears` to the other soil types, having a FALSE on the map variable `Peat`. The **spread** function calculates a total spreading time map from the locations occupied with the plant on Ini, multiplying the distance between cells and the 'friction' values on `Years`. The last operation in the initial section assigns a Boolean TRUE to all cells colonized within 50 years. The sequence of operations in

Table 2. Script for modelling groundwater flow, with remarks behind #.

```
preloop
  Dem='dem.map'                  # topographical elevation (m)
  FlowCond=''cond.map''          # flow conditions, flow or fixed head
  H=''hini.map''                 # head at start of model run (m)
  T=scalar(0.1)                  # timestep (days)
  EvapMax=0.001                  # maximum evapotranspiration (m/day)
  S=0.2                          # specific yield
  Rain='rainmeas'                # stack of maps with measured
                                 # rain per time step

initial (nrloops=500)
  # hydraulic conductivity m/day
  K=exp(2.4957+spatcor(0.2Nug(0) + 0.8 Exp(750)))

  dynamic (nrtimesteps=1600)
    # depth of water table (m)
    WTDepth=Dem-H
    # head in root zone
    Wet=WTDepth<0.45
    # evapotranspiration (m/day)
    Evap=max(EvapMax-0.001*WTDepth,0)
    # recharge (m/day)
    Recharge=max(Rain-Evap,0)
    # head (m)
    Head=transient(Head,Recharge*T,K,FlowCond,S,T)

postloop
  # duration of groundwater in root zone (days)
  Dur=timetotal(Wet,1)/T
  # duration of groundwater in root zone, 5% percentile
  Dur5P=sampleperc(Dur,5)
  # duration of groundwater in root zone, 95% percentile
  Dur95P=sampleperc(Dur,95)
  # significantly above threshold duration of 150 years
  AboveTh=Dur5P>150
  # significantly below threshold duration of 150 years
  BelowTh=Dur95P<150
```

the initial is executed 500 times, representing 500 Monte Carlo loops (samples), resulting in 500 realizations of the map variable `Col`. The operation in the postloop section derives from these 500 realizations of `Col`, for each cell the probability that it is occupied by the plant after 50 years, resulting in the map variable `ColProb` (figure 4(*c*)).

5.2 *Two-dimensional spatial and temporal stochastic model*

An example of a temporal model with error propagation is given in table 2. It concerns a hypothetical two-dimensional groundwater flow model meant to estimate the depth of the groundwater level below the surface, and the number of days that the groundwater level is in the root zone, which is assumed to be vulnerable for the vegetation. Figure 5 gives the most important map variables used in the model. The **preloop** section reads constant data from the database. These include the digital elevation model of the area, with elevation differences of only a few metres. The transmissivity is the only model input that is not exactly known, but

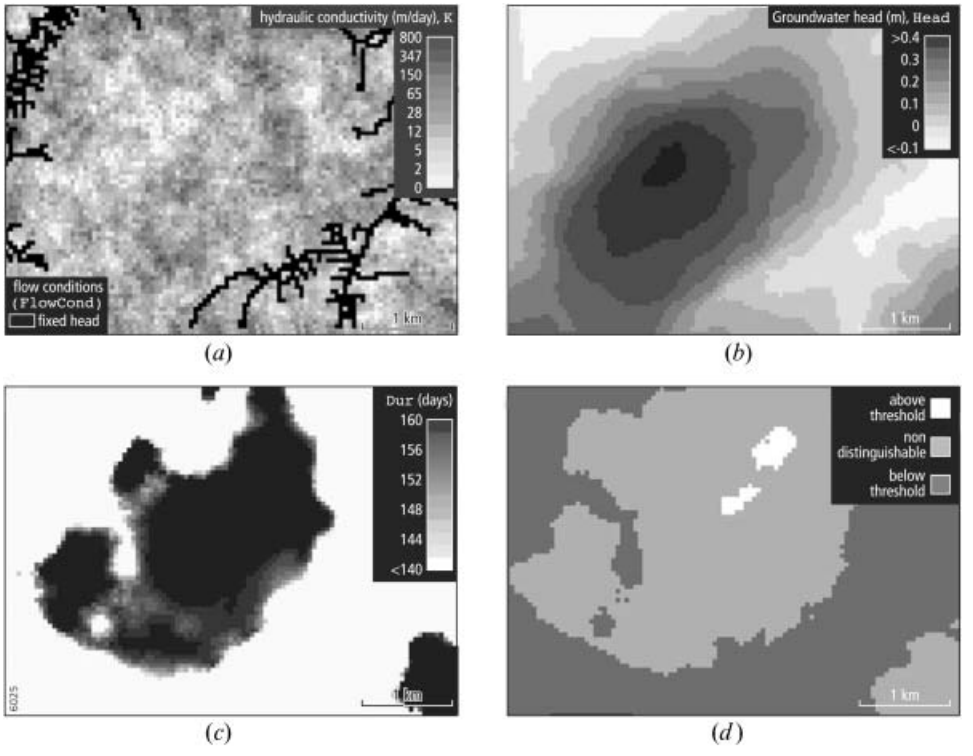


Figure 5. Groundwater flow model. (*a*) Realization of the hydraulic conductivity (grey scales, `K` in table 2, m day^{-1}), and flow conditions (`FlowCond`, black: river cells with a fixed head, other: flow). (*b*) Realization of the groundwater head (`Head`, metres) at the last time step. (*c*) Realization of the duration of groundwater in root zone (`Dur`, days). (*d*) Areas with duration of groundwater in root zone significantly below or above threshold duration (150 years), and areas for which it is not possible to determine whether the critical level is exceeded or not. Combination of `AboveTh` and `BelowTh` in table 2.

its statistical properties are known. As such, it is the only source of error or uncertainty in the model input, and it is modelled as a two-dimensional random field with spatial autocorrelation, described by a variogram. For each Monte Carlo loop, the **spatcor** function in the initial section generates a realization of this random field, using a combined nugget and exponential semivariogram, with a spatial autocorrelation range of 2000 m. The **exp** function calculates the base *e* exponent of this random field, resulting in a log-normal distribution of hydraulic conductivity (figure 5(*a*)).

The **dynamic** section is run for 1600 time steps, with a length of a time step of 0.1 days, explicitly defined with the map variable `T` in the **preloop** section. The **transient** function at the bottom of the dynamic section simulates transient groundwater flow in an unconfined layer (Crank-Nicolson method, Wang and Anderson 1982). For each time step, it calculates a map with groundwater heads (`Head`, figure 5(*b*)), derived from its input maps read from the database in the preloop section, the groundwater recharge (`Recharge`, m), the hydraulic conductivity (`K`, m day^{-1}), the flow conditions (`FlowCond`, figure 5(*a*)), the specific yield (`S`, dimensionless), and the time step (`T`, days). The first function in the **dynamic** section calculates the depth

of the water table below the soil surface, for each time step. The second function creates a map with a Boolean TRUE if the ground water level is closer than 0.45 m below the surface, assumed to be the root zone, otherwise a FALSE. The third function calculates the evapotranspiration as a function of the depth of the water table. The resulting map variable (`Evap`) is used in the fourth operation to calculate the recharge to the groundwater. The **postloop** section aggregates over time and over the Monte Carlo dimension. The **timetotal** function calculates the total duration (days) that the groundwater level has been in the root zone, for each Monte Carlo sample, resulting in the map variable `Dur` (figure 5(*c*)). The next two lines calculate the 5 and 95 percentiles of `Dur`, for each grid cell, using the **sampleperc** function. These percentiles define the lower and upper boundary of the 90% confidence interval. For each cell, the probability is 0.90 that the duration of the groundwater in the root zone lies between the `Dur5P` value and the `Dur95P`. By comparing these lower and upper boundaries of the confidence interval with a critical duration of 150 days, maps can be made where the duration of water in the root zone is certainly above (`AboveTh`) or below (`BelowTh`) the critical duration, at a confidence level of 90% (figure 5(*d*)).

5.3 *3D spatial and temporal model*

The first paper (Karssenberg and de Jong 2005) describes a three-dimensional model simulating deposition and erosion in an environment dominated by rivers. It was assumed that all model inputs were exactly known, which is not the case. Table 3 shows an extension to the script given in the first paper, which includes one additional (hypothetical) input associated with uncertainty: local-scale deposition. It is assumed that the total deposition (`Add`, m/time step, see table 3) for each time step is equal to the sum of deposition determined by large-scale processes (`DepCB`, m/time step) plus deposition determined by small-scale processes (`DepLocal`, m/time step). The large-scale deposition is modelled in the same way as in the first paper, as a function of distance to the channel belt. The small-scale deposition is represented by a stochastic variable with a (hypothetical) average value of 0.05, with a spatial pattern defined by a pure nugget variogram with a nugget variance of 0.01, for which realizations are drawn with the **spatcor** function, also described in section 5.2. This is done independently for each time step, which means that the pattern of local scale deposition is different for each time step.

Apart from the lines representing the deposition, the dynamic section of the script in table 3 corresponds to that given in the first paper. The **nrloops** keyword in the **initial** section sets the number of Monte Carlo loops to 500, which means that the dynamic model, defined in the dynamic section and consisting of 20 time steps, is run 500 times, resulting in 500 realizations of all model variables, for all time steps. As a result of the uncertainty in the deposition, each realization results in a different pattern of channel belts (not shown). The effect of this uncertainty is calculated in the **postloop** section, by calculating for each voxel the probability of occurrence of channel belt deposits, stored in the block variable `BeltProbBlock`. A two-dimensional picture of this block variable is given in figure 6, showing well-distinguished zones of low and high probabilities of occurrence of channel belt deposits. The location of the zones with high probability is determined by a fixed pattern of elevation values on the initial elevation model (`Dem`) and the processes included in the model.

Table 3. Script for modelling alluvial architecture, with remarks behind #.

```
preloop
  Depos='depos.map'                    # map with deposition rate at
                                       # the channel belt (m/timestep)
  In='inflow.map'                      # inflow location
  Dem='dem.map'                        # initial elevation model

initial(nrloops=500)

  dynamic (nrtimesteps=20)
    # create a local drain direction map
    Ldd=lddcreate(Dem)
    # centre of the channel belt,
    # path downstream from the inflow point
    Centre=path(Ldd,In)
    # distance to the channel belt centre line (m)
    Dist=spread(Centre,0,1)
    # channel belt with width of 1500 m
    Belt=Dist<750
    # deposition as a function of distance
    # to channel belt (m/timestep)
    DepCB=if(Belt then Depos else (Depos*exp((750-Dist)/1500)))

    # deposition caused by local scale processes (m/timestep)
    DepLocal=0.05+spatcor(0.01Nug())
    # total deposition (m/timestep)
    Add=DepCB+DepLocal

    # erosion and deposition (m/timestep, maps)
    Erosion=if(Belt then 10 else 0)
    Deposition=if(Belt then 10+Add else Add)

    # adjust thickness of block
    LithBlock=remove(Erosion)
    LithBlock=add(Deposition,Belt)

    Dem=top()

 postloop
  # probability of a channel belt
  BeltProbBlock=sampleprob(LithBlock)
```

7. Discussion and conclusions

Although the example models demonstrate that modelling error propagation is possible using the language, many improvements need to be made for a wider and better application. Compared with the language described in the first paper (Karssenberg and de Jong 2005), the amount of data that have to be managed by this language is even larger, because an additional Monte Carlo dimension has been added. Also, the sections in the script are executed in a fixed order, from top to bottom. This results in large amounts of (sometimes) unused data stored on hard disk, while calculation times may become excessively high. Optimization algorithms need to be developed that manage data storage, restricting data storage to (1) variables for which the model builder explicitly defines in the script that they need to be stored, and (2) those variables, time steps or Monte Carlo samples for which the results need to be stored since they are needed in a later phase script execution. In addition, algorithms need to be developed that find the optimal order of calculation

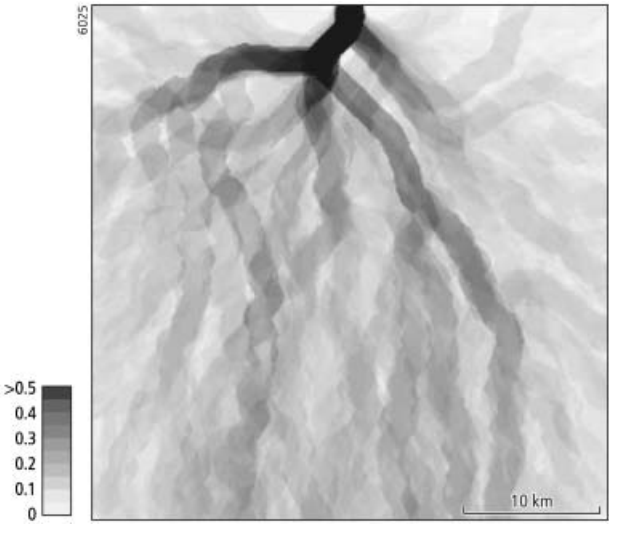


Figure 6. Alluvial architecture model, probability of occurrence of a channel belt, `BeltProbBlock` in table 3. Plan view of the block variable `BeltProbBlock`, showing the highest probability found below each x,y location.

of dynamic models in a Monte Carlo simulation scheme, e.g. whether the time steps should be nested in the loops of the Monte Carlo simulation or vice versa. The optimization schemes regarding data storage and calculation order need to parse the whole script before it is executed.

The multidimensionality of the data dealt with also has a major impact on the visualization routines needed to explore model inputs and outputs. Exploratory data analysis techniques and software for temporal, multidimensional data associated with environmental models hardly exist. It might be necessary to apply visualization techniques that go beyond these known in cartography or standard statistics, such as glyphs, which are graphical objects whose elements (e.g. position, size, shape, colour, orientation) are bound to data (Foley and Ribarsky 1994).

As noted in the first paper, the PCRaster research and development team will release a PCRaster Python extension based upon the prototype language described here. Information is available at http://pcraster.geo.uu.nl

Acknowledgements

We wish to thank Edzer Pebesma, Peter Burrough, Willem van Deursen, Marc Bierkens, and Cees Wesseling for providing many inputs to the research described here. This research was supported in part by EC grant Eurodelta, EVK3-CT2001-20001.

References

BIERKENS, M.F.P. and BURROUGH, P.A., 1993, The indicator approach to categorical soil data. I. Theory. *Journal of Soil Science*, **44**, pp. 361–368.

CROSETTO, M. and TARANTOLA, S., 2001, Uncertainty and sensitivity analysis: tools for GIS-based model implementation. *International Journal of Geographical Information Science*, **15**, pp. 415–437.

DAVIS, T.J. and KELLER, C.P., 1997, Modelling uncertainty in natural resource analysis using fuzzy sets and Monte Carlo simulation: slope stability prediction. *International Journal of Geographical Information Science*, **11**, pp. 409–434.

DEUTSCH, C.V. and JOURNEL, A.G., 1998, *GSLIB Geostatistical Software Library and Users's guide*, 2nd edition (New York: Oxford University Press).

FISHER, P.F., 1991, Modelling soil map-unit inclusions by Monte Carlo simulation. *International Journal of Geographical Information Systems*, **5**, pp. 193–208.

FOLEY, J. and RIBARSKY, B., 1994, Next-generation data visualisation tools. In *Scientific Visualisation. Advances and Challenges*, L. Rosenblum, et al. (Eds), pp. 103–126 (London: Academic Press).

GOODCHILD, M.F., GUOQING, S. and SHIREN, Y., 1992, Development and test of an error model for categorical data. *International Journal of Geographical Information Systems*, **6**, pp. 87–104.

HAMMERSLEY, J.M. and HANDSCOMB, D.C., 1979, *Monte Carlo Methods* (London: Chapman & Hall).

HEUVELINK, G.B.M., 1998, Error Propagation in Environmental Modelling with GIS (London: Taylor & Francis).

HEUVELINK, G.B.M. and BURROUGH, P.A., 2002, Developments in statistical approaches to spatial uncertainty and its propagation. *International Journal of Geographical Information Science*, **16**, pp. 111–113.

KARSSENBERG, D. and DE JONG, K., 2005, Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. *International Journal of Geographical Information Science*, **19**, pp. 559–579.

KIIVERI, H.T., 1997, Assessing, representing and transmitting positional uncertainty in maps. *International Journal of Geographical Information Science*, **11**, pp. 33–52.

LANTER, D.P. and VEREGIN, H., 1992, A research paradigm for propagating error in layer-based GIS. *Photogrammetric Engineering & Remote Sensing*, **58**, pp. 825–833.

LEUNG, Y. and YAN, J., 1998, A locational error model for spatial features. *International Journal of Geographical Information Science*, **12**, pp. 607–620.

NCDCDS, 1988, The proposed standard for Digital Cartographic Data. *The American Cartographer*, **15**, pp. 11–140.

PEBESMA, E. and WESSELING, C.G., 1998, Gstat: a program for geostatistical modelling, prediction and simulation. *Computers & Geosciences*, **24**, pp. 17–31.

PEBESMA, E.J., KARSSENBERG, D. and DE JONG, K., 2001, The stochastic dimension in a dynamic GIS. In *Compstat 2000, Proceedings in Computational Statistics*, J.G. Bethlehem and P.G.M. Van der Heijden (Eds), pp. 379–384 (Heidelberg: Physica-Verlag).

PRESS, W.H., TEUKOLSKY, S.A., VETTERLING, W.T. and FLANNERY, B.P., 1986, *Numerical Recipes in Fortran 77, the Art of Scientific Computing*, 1st edition (Cambridge: Cambridge University Press).

THAPA, K. and BOSSLER, J., 1992, Accuracy of spatial data used in Geographic Information Systems. *Photogrammetric Engineering & Remote Sensing*, **58**, pp. 835–841.

VEREGIN, H., 1994, Integration of simulation modeling and error propagation for the buffer operation in GIS. *Photogrammetric Engineering & Remote Sensing*, **60**, pp. 427–435.

VEREGIN, H., 1995, Developing and testing of an error propagation model for GIS overlay operations. *International Journal of Geographical Information Systems*, **9**, pp. 595–619.

WANG, H.F. and ANDERSON, M.P., 1982, *Introduction to Groundwater Modeling. Finite Difference and Finite Element Methods* (New York: Academic Press).