Towards improved solution schemes for Monte Carlo simulation in environmental modeling languages

Derek Karssenberg and Kor de Jong

Department of Physical Geography, Faculty of Geosciences, Utrecht University, PO Box 80115, 3508 TC Utrecht, the Netherlands. Tel. +31 30 2532768, Fax +31 30 2531145,
Email: d.karssenberg@geo.uu.nl, http://pcraster.geo.uu.nl.

1.1. Introduction

Numerical environmental models simulating landscape changes through time with equations representing physical, chemical, or biological landscape processes (Karssenberg & De Jong, 2005; Wainwright & Mulligan, 2004) are important tools for environmental research, planning, and management. Although numerical environmental models have much in common with the group of software tools known as Geographical Information Systems (GIS, Burrough & McDonnell, 1998) – both environmental models and GIS deal with spatial data – there are at least two large differences. The first difference is in the number of dimensions represented. Most GIS systems restrict their data models to two spatial dimensions, whereas numerical environmental models need to deal with at least five data dimensions: three spatial dimensions, the time dimension, and a dimension to represent uncertainty in a Monte Carlo framework. The second difference between GIS and environmental models is the type of functions on the data required. While GIS is strong in functions for static analysis, management and presentation of vector and raster data, numerical environmental models need to incorporate functions representing spatio-temporal processes occurring in a landscape such as numerical solution schemes of differential equations. These functions need to deal with additional dimensions such as the time dimension.

As a result of these two differences, run times of numerical environmental models are often much larger than these of GIS applications, since numerical environmental models include relatively complicated functions on data volumes which are often even larger than in most GIS applications. So, minimizing run times is a very relevant issue when coding environmental models. This requires specialist knowledge in the disciplines of informatics, computational geometry, and numerical mathematics. Since model builders, which are environmental scientists with knowledge of landscape processes, often do not have this knowledge, model construction is preferably done with high level environmental modeling languages (Karssenberg, 2002). These are languages providing building blocks for model construction with built-in optimization schemes developed by specialists in the abovementioned disciplines. The environmental scientists construct their models by combining these building blocks instead of programming everything from scratch. An example of such a language is the PCRaster language (PCRaster, 2006) and the recently developed PCRaster Python library (Karssenberg & De Jong, subm.). Concepts and examples presented in this paper are taken from these languages although many other environmental modeling languages exist with similar concepts (c.f., Karssenberg, 2002).

This paper starts with a description of the multiple dimensions involved in numerical environmental modeling and functions required operating on data in these dimensions. The second part of the paper discusses future challenges for designing better stochastic environmental modeling languages that minimize runtimes involved in calculating parameters describing the sampling distributions of output variables generated by Monte Carlo simulation.

1.2. Stochastic dynamic spatial modeling

1.2.1. Process representation, discretisation of temporal and spatial dimensions

To mimic changes in a landscape through time, numerical environmental models use the rules of cause and effect, with a discretisation of time in time steps. The state of the model variables, which are attributes in one, two, or three dimensional space, at time t+1 is defined by their state at t and a function f. Similar descriptions can be found in (Beck, Jakeman, & McAleer, 1993; Karssenberg & De Jong, 2005; Wainwright & Mulligan, 2004):

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

The model variable(s) $Z_{1..m}$ belong to coupled processes, and therefore have feedback in time, for instance stream water level. The model variable(s) $I_{1..n}$ are simple inputs to the model, for instance incident rainfall in a runoff model. The function f with associated parameters $P_{1..l}$ models the change in the state of all model variables over the time step t to t+1 and it can be either an update rule, explicitly specifying the change of the state variable over the time slice (t, t+1), for instance a rule based function such as cellular automata (e.g., Toffoli, 1989), or alternatively a derivative of a differential equation describing the change of the state variables as a continuous function (c.f. Wainwright & Mulligan, 2004). It may also include probabilistic rules when model behaviour is better described as a stochastic process. The function f is evaluated for each time step, which can be written in pseudo code as:

```
for each time step:                                    (2)
  evaluate f
```

To represent spatial variation, the model variables are attributes in two or three dimensional space, referred to as map variables or block variables, respectively (Figure 1). For representing continuous fields, most packages for environmental modeling discretise the lateral dimensions in regular grid cells. PCRaster discretises the third dimension with an irregular discretisation (ragged array) using voxels with variable thickness (Figure 1, Karssenberg & De Jong, 2005).

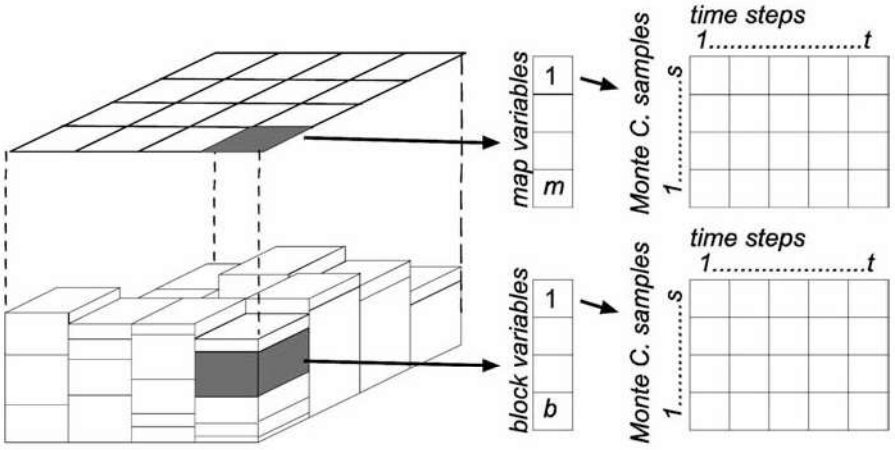

Figure 1. Representation of spatial dimensions, time, and Monte Carlo samples in environmental modeling. Left, map and block; centre, list of map or block variables; right, matrix with values for each time step and each Monte Carlo sample, stored for each map or block variable, for each cell or voxel.

To represent the change in state of a model over each time step, environmental modeling languages come with a library of preprogrammed functions on map and block variables. These functions may represent spatial interaction such as required in cellular automata models, lateral flow over a digital terrain model or simple processes without spatial interaction (c.f. Karssenberg & De Jong, 2005). The preprogrammed functions are the building blocks of the model: the model builder represents f in equation (1) by combining the functions in an iterative script. To illustrate this, consider the construction of a rainfall-runoff model that simulates surface runoff of water as a result of incident rainfall. The spatial variation in rainfall, runoff, infiltration, and other state variables is represented by using map variables. The processes involved generating surface runoff are programmed by combining functions on maps, in order to represent f in equation 1. By running these functions for each time step, a simulation can be made of the temporal change in rainfall, runoff, and related processes.

1.2.2. Stochastic modeling with Monte Carlo simulation

In many cases, environmental models include probabilistic rules or have inputs or parameters that are given as spatial probability distributions as is the case in error propagation modeling (c.f. Crosetto & Tarantola, 2001; Heuvelink, 1998). The aim of stochastic modelling is to derive the probability distributions (or parameters describing these) of the stochastic model variables $Z_{1..n}(t)$ from the stochastic inputs, parameters, or probabilistic rules, 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 only be approximated with Monte Carlo simulation modelling (Heuvelink, 1998). Monte Carlo simulation involves two steps: Step (1). For each Monte Carlo sample: run f(·) (eq. 1) for all time steps with realizations of stochastic parameters or inputs, and store the realizations of the model output variables $Z_{1..n}$ in which the interest lies. Step (2). Compute descriptive statistics (e.g., mean, variance, quantiles) from the model outcomes of each sample, for each model output variable and each time step t, or for a selection of model variables and time steps. In pseudo code, Monte Carlo simulation for environmental models can be written as (see also Figure 2):

```
1     for each sample:
        for each timestep:                                    (3)
3          evaluate f
      compute descriptive statistics of output variables
```

This nested iteration will provide each variable and each cell or voxel with a value for each time step and Monte Carlo sample, which means that the array on the right side of Figure 1 is filled. In Monte Carlo simulation, the calculation of descriptive statistics typically involves an aggregation over the samples of the Monte Carlo dimension: for each cell or voxel, a statistical value such as the mean or a quantile is calculated from the values of the Monte Carlo samples for that cell or voxel (Figure 3C). The example presented in the previous section will make this clear. Consider the rainfall-runoff model predicting the change in surface runoff through time, for each grid cell. It is run in Monte Carlo mode now to represent uncertainty in incident rainfall. Each Monte Carlo sample represents a realization of the model with a possible spatial distribution of runoff. To calculate the uncertainty in the surface runoff for each time step and each cell, the model variable surface runoff is aggregated over the Monte Carlo dimension, by calculating for instance the variance of the surface runoff values over the Monte Carlo dimension. This is done for each cell and each time step.

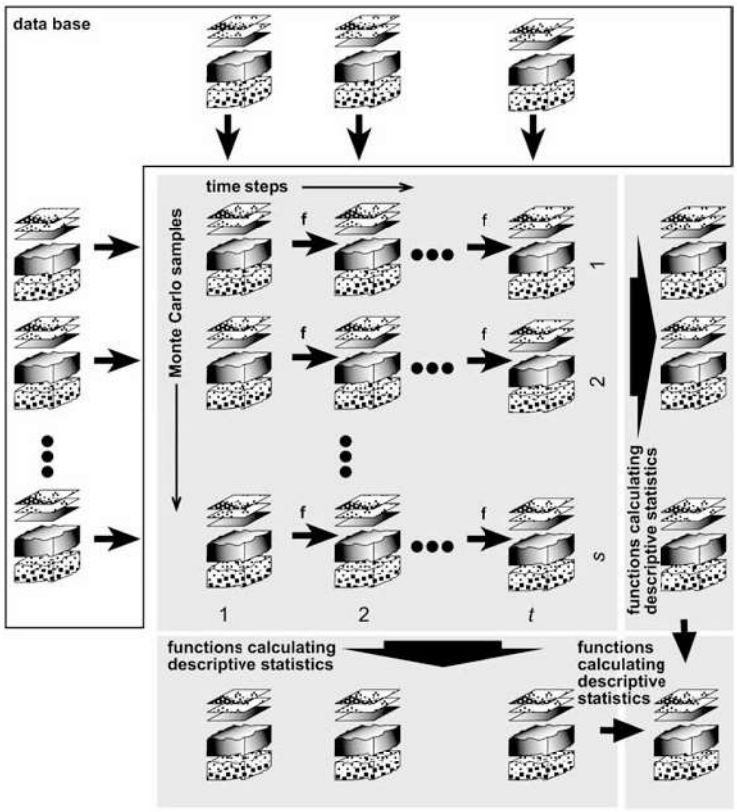

Figure 2. Concepts for temporal, two or three dimensional, and stochastic modeling.

In addition to the calculation of descriptive statistics over the Monte Carlo dimension, descriptive statistics can be calculated over time and/or over space. Consider for instance the calculation of peak runoff in the example rainfall-runoff model. This involves an aggregation over time (Figure 2B) where the maximum runoff is calculated over the time steps, for each cell and each Monte Carlo loop. Calculating average runoff values for different land use types is an example of aggregation over space (Figure 2A), where average values need to be calculated over a certain set of cells, for each time step and each Monte Carlo loop. Finally, aggregation needs to be done in certain cases over more than one dimension. Consider for instance the calculation of the median peak runoff, which involves an aggregation over time, calculating the peak runoff for each Monte Carlo sample, and an aggregation over the Monte Carlo dimension, calculating the median of the peak runoff values over the Monte Carlo loops.

The development of environmental modeling languages with built-in functionality for Monte Carlo simulation is still in its infancy. Recently, the PCRaster team developed the PCRaster Python library (Karssenberg & De Jong, subm.) which can be considered as one of the first languages that can do Monte Carlo simulation with dynamic spatial environmental models.

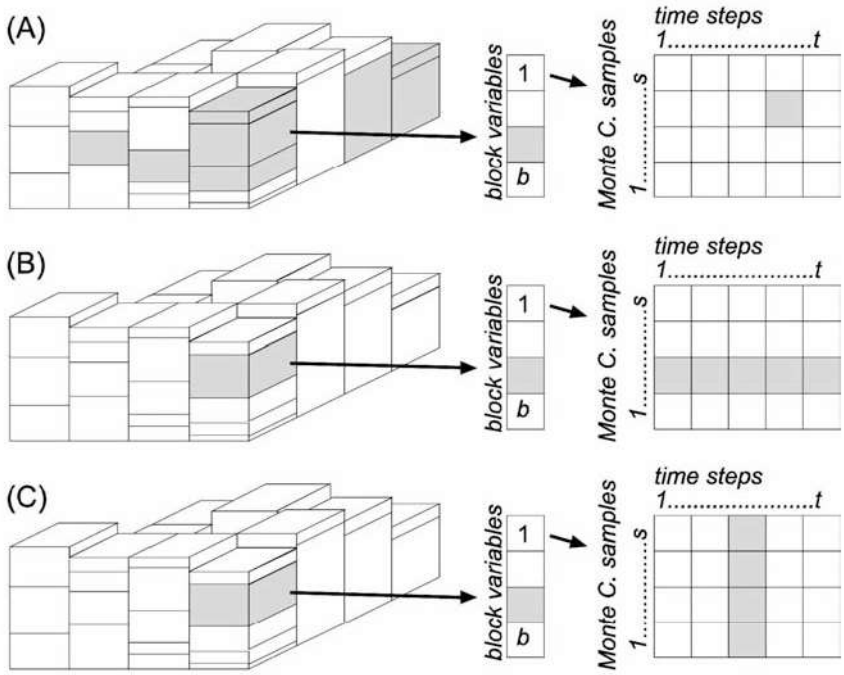

Figure 3. Functions calculating descriptive statistics, aggregating over A) space, B) time, C) the stochastic dimension, i.e., Monte Carlo samples.

1.3. Minimizing run times involved with Monte Carlo simulation

1.3.1. Examples of solution schemes reducing I/O

As noted in the introduction, the run times of environmental models can be large due to the relatively large number of calculations involved in evaluating f (equation 1), the large data sets used, and the large number of evaluations of f required, which is equal to the number of time steps times the number of Monte Carlo loops. Although decreasing run times of f is still a major challenge when designing new environmental modeling languages, we will focus our discussion here on the run times associated with the calculation of descriptive statistics. Unlike run times of f being mainly dependent on the speed of the processor, run times associated with calculating descriptive statistics are largely dependent on memory, since limited memory or inefficient use of memory may require I/O which will increase the run times significantly. An example will make this clear. A prototype implementation of the environmental modeling language known as the PCRaster Python Library (Karssenberg & De Jong, subm.) uses the following order of calculation for calculating descriptive statistics over the Monte Carlo dimension of a map variable.

```
1     for each sample:
2       for each time step:
3         evaluate f
4         write map variable to disk
5     for each cell:                                    (4)
6       for each time step:
7         for each sample:
```

```
8           read cell value from disk and store in memory
9         compute descriptive statistics of variable
10        release cell values from memory
```

This approach writes the map variable for which statistics need to be calculated to hard disk (line 3 in the scheme above), storing the map variable as a separate file for each time step and each Monte Carlo sample. In line 5-10, these files are opened to calculate descriptive statistics of the variables: the scheme aggregates for each time step over the samples, one cell at a time. Since the calculation is done cell-by-cell, a file needs to be opened, read from, and closed for each cell, Monte Carlo sample and time step. This means that a file is opened a number of 10^7 up to 10^{15} times, which is calculated as $t \cdot s \cdot c$, with t, number of time steps (typically 10^1-10^4), s, number of samples (10^2-10^3), c, number of cells or voxels per variable (10^4-10^8). The advantage of this approach is that the number of cells, time steps and Monte Carlo samples is not limited by the memory available, since the amount of memory required equals $s \cdot m$, which is in the order of magnitude of 10^2 byte – 1 kB using the typical values given above and a memory per cell/voxel value (m) of 4 byte. The main disadvantage of this approach is the long run times since much time is spent on I/O.

A significant decrease of run times is reached when all data required for calculating statistics are kept in memory. Since an extensive evaluation of all possible approaches is beyond the aim of this paper, I provide two obvious approaches here as an invitation to the discussion below. The first one nests the time step loop inside the loop over the Monte Carlo samples:

```
1     for each sample:
2       for each time step:
3         evaluate f                                              (5)
4         keep map variable in memory
5     for each time step:
6       compute descriptive statistics of variable
7       release cell values from memory for current time step
```

This approach aggregating for each time step over the samples, one map at a time, requires a memory of $t \cdot s \cdot c \cdot m$ which is 10 MB up to 100 TB (tera byte, 10^{12} Byte) using the typical values given above. In practice, most desk top computers will not have sufficient memory for this approach, although it will work when the data in one or more dimensions is small. An alternative approach would be to do the loop over the Monte Carlo samples nested inside the loop over the time steps:

```
1     for each time step:
2       for each sample:
3         evaluate f                                              (6)
4         keep map variable in memory
5       compute descriptive statistics of map variable
6       release values from memory for current time step
```

This approach requires much less memory, equal to $s \cdot c \cdot m$, which is typically 1 MB – 100 MB.

1.3.2. Factors involved in choosing optimal solution schemes

The examples in the previous section show that the straightforward solution scheme (4) currently applied in the PCRaster Python Library is not in all cases the best scheme considering calculation time. In many cases, other schemes can be applied that do not involve I/O speeding up the calculation. It is a major challenge for future research to develop better environmental

modeling languages that select the best approach for calculating descriptive statistics given the model script developed by the model builder, the size of the input data, and the properties of the computer used. As a first step towards such a 'clever' environmental modeling language, I discuss below the main factors which are important in selecting the best approach for calculating descriptive statistics, whereby focus is on schemes that minimize I/O.

Type of descriptive statistics required. In the solution schemes (4-6) it was assumed that the calculation of a value describing the distribution of a variable requires that all data reside in memory. Although this is required for calculating the exact value of quantiles, this is not needed for the calculation of statistics such as mean or variance since these can be calculated on-line (incrementally). On-line calculation means that the statistical value is repetitively updated using single data values which are sequentially read and released from memory. When descriptive statistics need to be calculated that allow for on-line calculation, solution scheme (6) can be adjusted:

```
1     for each time step:
2       for each sample:
3         evaluate f                                                    (7)
4         keep map variable in memory
5         update descriptive statistics of map variable
6         release values from memory for current time step and sample
```

The size of the memory required in this scheme is $c \cdot m$, typically 10 KB - 100 MB, which is orders of magnitude smaller than the memory required for solution scheme (6). This example shows that the type of descriptive statistics that needs to be calculated is an important factor in choosing the solution scheme. In general, on-line methods for calculation of statistics will be preferable. When exact calculation of a certain statistic is not possible with an on-line method, it should be considered to include on-line methods that return approximations of the statistic (e.g., Pearl, 1981) in environmental modeling languages.

Size of the data in each of the dimensions. The memory required for each of the solution schemes depends on the number of time steps and Monte Carlo samples, the number of cells or voxels, and/or the number of variables for which descriptive statistics need to be calculated. Certain schemes are possible (and very fast) with a small number of data, while the same schemes may become impossible with large data sets due to memory overflow. Note that in many cases descriptive statistics are not required for all coordinates in all dimensions. For instance, calculating surface runoff at the outflow point of a catchment requires the calculation of descriptive statistics at a single cell, the cell corresponding to the outflow point, only. Other cells can be ignored in the solution scheme which will decrease the amount of memory required.

Dimensions over which aggregation is required. In error propagation modeling, calculation of descriptive statistics is in most cases required over the Monte Carlo dimension on a cell-by-cell basis, resulting in descriptive statistics for each time step and each cell (or voxel) as discussed so far. Sometimes, it may be required to calculate statistics over the spatial or temporal dimension, too. Aggregation over the spatial dimension require relatively little memory in the order of magnitude of $c \cdot m$ since it can be done directly after evaluating the function f:

```
1     for each sample:
2       for each time step:                                             (8)
3         evaluate f, keep map variable in memory
4         calculate spatial stats of variable and release from memory
```

Calculation of descriptive statistics over the time dimension can be done following (8) when calculation can be done on-line, for instance calculating the maximum value of runoff over the time steps, for each cell. When descriptive statistics over the time dimension need to be calculated that require to have all data in memory (e.g., quantiles), the map variable needs to be kept in memory over all time steps, requiring $t \cdot c \cdot m$ of memory:

```
for each sample:
  for each time step:                                          (9)
    evaluate f, keep map variable in memory
  calculate spatial stats of variable and release from memory
```

Occurrence of multiple variables for which aggregation is required. In many cases, aggregation needs to be done for a number of different variables. Apart from increasing the total memory needed, this may complicate the solution schemes since conflicting solution schemes may be required for different variables. For instance, aggregation of a variable over the time dimension may require solution scheme (8), while aggregation over the Monte Carlo samples may require scheme (7). In this case, a solution scheme needs to be chosen by comparing efficiency of different possible schemes.

Occurrence of corresponding values. In some cases, a map or block variable may contain cells with exactly similar starting values and process equations, resulting in similar cell values over one, or multiple, dimensions. For instance, the amount of infiltration may be similar in all cells in a certain soil class, and as a result all cells in that class will have similar cell values. In this case, the calculation of descriptive statistics can be restricted to a single calculation for each soil class, instead of calculating statistics for each individual cell in the class. This principle can be used to decrease the amount of memory required in calculating descriptive statistics.

Memory requirements. In cases when the solution scheme requiring the smallest amount of memory results in a required memory that is larger than the available memory, a scheme needs to be applied that does I/O. Solution schemes need to be developed that result in shortest run times in this situation.

Memory required for the process model. In addition to the memory required for calculating descriptive statistics, the evaluation of the function f for each time step may need a large amount of memory, both for the evaluation of f itself and for storing variables at the end of each time step that are required as input to f for the next time step. So, by comparing different solution schemes, the amount of memory involved in the evaluation of f for each possible scheme needs to be considered, too. For instance, solution scheme (6) may be very efficient in terms of memory use related to calculating descriptive statistics, but it may not be as efficient as solution scheme (5) when memory is considered required for evaluating f. The point is that in scheme (5) all variables that need to be stored at the end of a time step as input to the next time step require to reside in memory for all Monte Carlo samples. This requires a memory in the order of magnitude of $s \cdot c \cdot m \cdot v$, with v, the number of variables that need to be stored.

1.4. Final remarks and conclusions

We showed that existing environmental modeling languages provide all functionality required for constructing models that simulate changes through time in two and three spatial dimensions. In addition, the PCRaster Python Library includes a framework for Monte Carlo simulation with these models, whereby functions can be used to calculate descriptive statistics of stochastic variables. Although this framework makes it much easier to do Monte Carlo simulation for environmental scientists compared to the situation whereby everything needs to be programmed

from scratch, the framework does not solve the problem of long runtimes associated with Monte Carlo simulation. It is a major challenge for the GIS research community to develop better environmental modeling languages that optimize, in terms of run times, the solution of stochastic models developed with these languages. The list of factors that need to be taken into account when choosing optimal solution schemes provided here should be considered as a first step towards such faster environmental modeling languages.

1.5. References

Beck, M. B., Jakeman, A. J., & McAleer, M. J. (1993). Construction and evaluation of models of environmental systems. In M. B. Beck, A. J. Jakeman & M. J. McAleer (Eds.), Modelling change in environmental systems. New York: John Wiley & Sons Ltd.

Burrough, P. A., & McDonnell, R. A. (1998). Principles of Geographical Information Systems. Oxford: Oxford University Press.

Crosetto, M., & Tarantola, S. (2001). Uncertainty and sensitivity analysis: tools for GIS-based model implementation. International Journal of Geographical Information Science, 15(5), 415-437.

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

Karssenberg, D. (2002). The value of environmental modelling languages for building distributed hydrological models. Hydrological Processes, 16(14), 2751-2766.

Karssenberg, D., & De Jong, K. (2005). Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. International Journal of Geographical Information Science, 19(5), 559-579.

Karssenberg, D., & De Jong, K. (subm.). Modelling landscape dynamics with Python. International Journal of Geographical Information Science.

PCRaster. (2006). PCRaster internet site. Retrieved jan 01, 2006, from http://pcraster.geo.uu.nl
Pearl, J. (1981). A space-efficient on-line method of computing quantile estimates. Journal of Algorithms, 2, 164-177.

Toffoli, F. (1989). Cellular automata machines. Cambridge, Massachusetts: MIT Press.
Wainwright, J., & Mulligan, M. (2004). Environmental Modelling. Chichester: Wiley.