A tool for construction of stochastic spatio-temporal models assimilated with observational data

Derek Karssenberg, Oliver Schmitz, Luis Manuel de Vries, Kor de Jong

Department of Physical Geography, Faculty of Geosciences, Utrecht University, Heidelberglaan 2, PO Box 80115, 3508 TC, Utrecht, the Netherlands. Email: d.karssenberg@geo.uu.nl

Abstract. A framework and a software tool has been developed to execute two important steps in the model development cycle of temporal numerical models simulating geographic change using rules of cause and effect. The first step supported by the tool is the software implementation of spatio-temporal models. This can be done with the tool by combining generic operations on 2D map and 3D block data in an iterative script structure included in the standard Python programming language. The second step supported by the tool is the optimisation of models using the methods of genetic algorithms and particle filters. A number of case study models is presented to illustrate how the tool works.

INTRODUCTION

Temporal numerical models simulating geographic change are one of the cornerstones of research in geography and earth science and are frequently used in management, planning, and risk assessment in application domains such as land use change (Ligtenberg *et al.* 2004, Moulin *et al.* 2004), hazards and evacuation (Helbing *et al.* 2000), ecosystem studies (Sydelko *et al.* 2001, Gimblett *et al.* 2003), spread of diseases (Breukers *et al.* 2006), criminology (Groff 2007), or hydrology (Giupponi *et al.* 2006). Although the application field may vary, temporal numerical models have in common that they simulate change over time using equations that represent real world processes (Wesseling *et al.* 1996, Burrough 1998), whereby the state of the modelled system at each moment in time is a function of its state in the past. Another common characteristic is that processes are modelled in a spatial-explicit way, which means that spatial patterns and spatial interaction in the system are taken into account (Karssenberg *et al.* 2005a).

As it is required to use models tailored to the research goals of a project, the data availability and the properties of the system being modelled (Karssenberg *et al.* 2006b), model construction is central in almost any research project that involves modelling. Two important steps in the model development cycle (Karssenberg *et al.* 2006b) are the conversion of the conceptual model structure to computer code, i.e. the implementation or construction of the model, and the calibration or assimilation of the model with real world spatio-temporal observational data collected by remote sensing, automatic data loggers, questionnaires, or retrieved from large data bases. Although both the construction of the model and data assimilation can be done by programming software from scratch using system programming languages, it is preferable to use tools at a higher level of abstraction that can be used by scientists and modellers without specialist knowledge in programming (van Deursen *et al.* 2000, Karssenberg 2002).

This paper describes such a tool which is developed by combining components of the PCRaster software for spatio-temporal modelling (van Deursen 1995, Wesseling *et al.* 1996, PCRaster 2006) and new modules for calibration using the genetic algorithm (Coley 1999) and data assimilation with the particle filter (van Leeuwen 2003, Weerts *et al.* 2006). The modeller has access to these components and combines them with the generic Python scripting language (Python 2007). The focus is here on simulation using continuous spatial fields. Similar approaches can be followed for generic

toolboxes that use discrete entities as modelling framework (e.g., Goodchild *et al.* 2007, Bithell *et al.* in press), although a number of modifications are required to the concepts described in this paper.

MODEL CONSTRUCTION

Spatio-temporal process based modelling

Process-based, or transient modelling, involves a discretisation of the time in time steps *t*. To represent processes and the evolution of the system over time, a functional f is applied for each time step:

For each time step *t*: (1)

$$\mathbf{z}(t+1) = f(\mathbf{z}(t), \mathbf{i}(t), \mathbf{p})$$

Here, $\mathbf{z}(t)$ is a set of state variables representing the situation or state of the system at time step *t*. These are 2D or 3D spatial fields. The change over the duration of a time step resulting in the state of the system $\mathbf{z}(t+1)$ at the next time step is represented by the functional f, which can be a simple update rule, a solution of a differential equation, or a cellular automata (Karssenberg *et al.* 2006b). Other inputs to f are the drivers of the system, which is the set of input variables $\mathbf{i}(t)$, and a set of parameters **p** of f. Note that all terms in the equation refer to 2D or 3D spatial entities, while f is a spatial function.

A tool for model construction needs to provide facilities to represent the spatial entities ($\mathbf{z}(t)$, $\mathbf{i}(t)$ and **p**), the functional f, and the iteration over the time steps. In our tool, the spatial entities are time series of 2D maps or 3D blocks with a spatial discretisation in cells and voxels, respectively. The modeller represents f for each time step by a combination of generic operations on these maps and blocks. These operations derive their concepts from map algebra and cartographic modelling (Tomlin 1990, Wesseling *et al.* 1996, Pullar 2001) and are grouped in point operations, neighbourhood operations, operations using topology between cells, and operations calculating statistics over areas or volumes (Karssenberg *et al.* 2005a). The model builder can combine these operations in an iterative Python script. This is accommodated by a specific Python model class that comes with the tool (Karssenberg *et al.* 2007). In this model class, the model developer can combine the generic operations on blocks and maps that are included as a Python module. Also, the class provides standard functions to read and write temporal map and block data to disk. Model construction using this toolset is done by combining high level building blocks that represent spatial processes. This has a number of advantages compared to model construction from scratch using a system programming language (Karssenberg 2002).

CALIBRATION AND DATA ASSIMILATION

Genetic algorithm

An important aspect of modelling is to find the correct representation of the real world processes in the model. The modeller does this by choosing the correct set of operations to represent f in equation 1. Equally important is finding the correct values of the inputs $\mathbf{i}(t)$ and the parameters **p**. Standard values are used for inputs and parameters, often derived from field measurements or literature. However, it is possible to further optimise these values by using inverse modelling, also referred to as calibration. In inverse modelling, inputs and parameters are adjusted by minimizing the difference between output variables of the model $\mathbf{z}(t)$, and observations or measurements $\mathbf{y}(t)$, at the time steps *t* for which observations are available. This difference is quantified in an objective function $g(\mathbf{z}(t),\mathbf{y}(t))$, which calculates for instance the mean square difference between the observations and the measurements, although many other functions are possible (Janssen *et al.* 1995). It is assumed that the best set of values for the inputs and parameters of a spatio-temporal model corresponds to the set of

these values resulting in the minimal value of the objective function (McLaughlin *et al.* 1996). Many different approaches to minimize the objective function exist, each with their advantages and disadvantages. The genetic algorithm is particularly efficient when the relation between the inputs/parameters and the objective function is very irregular (Bennett *et al.* 1998, Coley 1999).

The idea behind genetic algorithms finds its roots in biology. The success of adaptation of species to all sorts of external changes has been used as inspiration to develop an analogous approach for model optimisation. The actual algorithm is a simplified representation of natural selection that nonetheless provides effective results. All the parameters and inputs to be optimised are encoded into one big string, called chromosome. Each model run with a specific set of input and parameter values is characterized by this chromosome and referred to as an individual. Multiple individuals together form a population. After running the model for all individuals, and the calculation of the objective function for each individual, a new generation of individuals is created on the basis of the fitness, i.e. the objective function value, of each individual. Individuals with a low fitness are removed while the remaining individuals may stay as they are, or receive small changes in their chromosome. New individuals are created by combining existing chromosomes to generate new chromosomes. The individuals in the new population, i.e. the next generation, perform better (provide a lower value of the objective function) and the fitness of the population has increased. The process above is repeated iteratively until an acceptable model result is achieved as follows:

For each generation g:

For each individual i:

For each time step t: (2)

$$\mathbf{z}_{g,i}(t+1) = \mathrm{f}(\mathbf{z}_{g,i}(t), \mathbf{i}_{g,i}(t), \mathbf{p}_{g,i})$$

Evaluate objective function

Update population

A number of generations g is run, typically 10-100. For each generation, a number of i model runs (i.e., individuals) is executed, each with its specific parameters and inputs stored in the chromosome of that model. After each model run, the objective function value is calculated. At the end of the evaluation of a generation, the generation is updated using the principles taken from biology as described above, resulting in a new generation.

The tool developed in our team provides facilities that make it relatively simple to optimise spatio-temporal models using this genetic algorithm. It provides the control flow to iterate over generations and individuals. Also, it has built in methods to update the generation such as selecting fittest individuals, changing chromosomes, and creating new individuals using existing chromosomes. Unlike other tools for genetic algorithms, it is specifically meant for spatio-temporal models and can optimise static variables, such as parameters, but also dynamic variables, like temporal inputs, whereby the input is optimised for each individual time step separately. It is fully integrated in the Python model class that is also used for construction of spatio-temporal models described in the first section. To reduce run times, the execution of the individuals can be distributed over different processors, if available, using a built-in forking mechanism.

In a benchmark study, the tool was used in an inverse modelling procedure of alluvial architecture models (Karssenberg *et al.* 2001). Alluvial architecture models are forward spatio-temporal models that mimic processes of deposition and erosion on a river plain, over time spans of 10^2-10^5 years, resulting in a 3D block of sediments. The two types of sediments created by these models are channel belt deposits, typically sandy deposits created by a meandering river; and overbank deposits, clay deposits deposited between rivers. The main focus is on the channel belt deposits because these are reservoirs for oil and water. In order to predict the 3D geometry of channel belt deposits in a study area located in the Rhine Meuse delta, the Netherlands, we used an alluvial architecture model in

combination with a data set of boreholes with depositional types collected in the study area. The inputs and parameters of the alluvial architecture model were optimised in order to fit the predicted 3D geometry of the channel belts with the observed sediments at the borehole locations. Both the forward alluvial architecture model and the genetic algorithm were run in the tool described above. Results are shown in Figure 1.

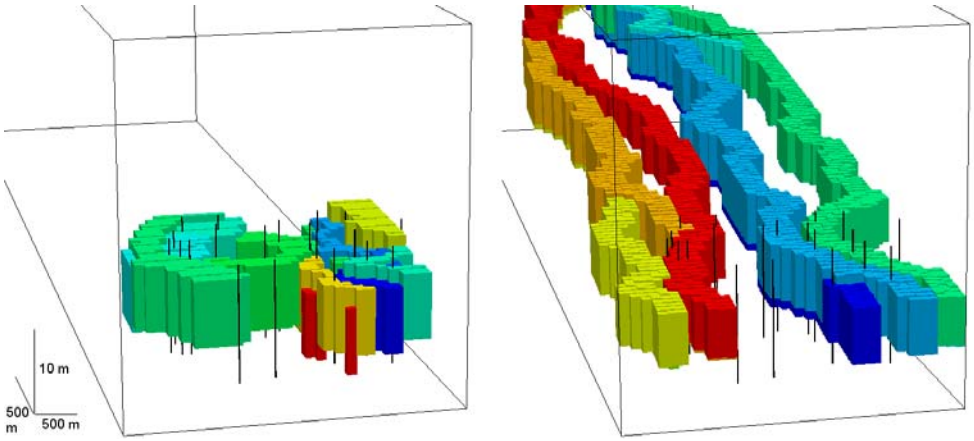

Figure 1. Geometry of channel belts. Each individual channel belt is represented by a different colour. Left: observed channel belts; right: modelled channel belts. Observed and modelled outcome correspond at borehole locations. Borehole locations are indicated with vertical bars. Scale on left side applies to both panels.

Particle filter

Another approach of improving model predictions using data is the use of filters. Unlike the Kalman filter, the particle filter has the advantage that it can be used with non-Gaussian distributions of variables while the conservation of mass in model equations is not violated as may be the case when Kalman filters are used (van Leeuwen 2003, Weerts *et al.* 2006). The particle filter uses a stochastic description of the system. The state variables, inputs and parameters in equation 1 become stochastic variables characterized by a probability density function. As geographic models are too complex to solved the functional f in an analytical way when its inputs are stochastic variables, these models are run using Monte Carlo simulation. The calculation order currently implemented in the tool for Monte Carlo simulation is (Karssenberg *et al.* 2005b, Karssenberg *et al.* 2006a):

For each Monte Carlo sample s:

For each time step t: (3)

$$\mathbf{z}_s(t+1) = \mathrm{f}(\mathbf{z}_s(t), \mathbf{i}_s(t), \mathbf{p}_s)$$

Aggregate over Monte Carlo samples

The spatio-temporal model is run for a number of so called Monte Carlo samples s. For each sample, a different set of realizations of stochastic inputs and parameters is used. So, in equation 3 above, $\mathbf{z}_s(t)$, $\mathbf{i}_s(t)$ and $\mathbf{p}_s$ refer to realizations of random variables associated with sample s. The realizations of all samples s together, for a certain time step, represent the probability density function of these entities for that time step. After executing the model for all Monte Carlo samples, statistical values need to be calculated characterizing the probability density function of the state variables. This aggregation is done for each time step (Karssenberg *et al.* 2005b).

Unlike the genetic algorithm, which optimises inputs and parameters of a model as a whole, for all time steps, the particle filter improves the state of the model at time steps when observational data are available, referred to here as filter moments. Should data be available for multiple time steps, the filter algorithm is run for each of these time steps. The filter algorithm improves the state of the model at a filter moment by calculating the values of the state variables using a Bayesian function which is a combination of the values calculated by the spatio-temporal model of equation 3, the prior distributions, and observational data. This results in a posterior distribution of the state variables at the time step of the filter moment which is used as input for the next time step of the spatio-temporal model.

This procedure is a standard component of the modelling tool which executes the following sequence of calculations:

For each period p:

For each Monte Carlo sample s:

For each time step t in the period: (4)

$$\mathbf{z}_{p,s}(t+1) = \mathrm{f}(\mathbf{z}_{p,s}(t), \mathbf{i}_{p,s}(t), \mathbf{p}_{p,s})$$

Execute the particle filter

Each filter moment is associated with a period p consisting of all time steps after the previous filter moment up to and including the time step of the respective filter moment itself. So, the number of periods equals the number of filter moments. For each period, the model is run in Monte Carlo mode. At the end of the period, after executing all time steps, the particle filter is applied which involves a removal or cloning of the Monte Carlo samples s (c.f., (van Leeuwen 2003, Weerts *et al.* 2006)). This procedure represents the Bayesian function to calculate the posterior distribution of state variables. A prototype of the tool with the particle filter is currently developed.

CONCLUSION AND FINAL REMARKS

A framework and tool for construction of spatio-temporal models has been developed that can be used to construct models in a wide range of application domains. Models constructed with this tool can be optimised using data with two different optimisation schemes included in the tool: a genetic algorithm and a particle filter. Most components of the tool will be distributed at http://pcraster.geo.uu.nl in 2008, free of charge. The particle filter is still under development and will be distributed later.

Acknowledgements

Hans van der Kwast (VITO, Belgium) is acknowledged for his extensive evaluation of an early prototype of the particle filter software. Kim Cohen (Department of Physical Geography, Utrecht University) provided many inputs to the alluvial architecture model case study. Cees Wesseling and Willem van Deursen (PCRaster Environmental Software) are thanked for their inputs in the development of the PCRaster Python software. This research was partly funded by ‘Ruimte voor Geo-Informatie’, project ‘On-line coupling of spatial optimisation tools and spatially distributed simulation models’, RGI 313. Acknowledgement is made to the Donors of the American Chemical Society Petroleum Research Fund for support of this research. Three anonymous reviewers are thanked for their comments to an earlier version of this paper.

BIBLIOGRAPHY

Bennett, D.A., Armstrong, M.P. and Wade, G.A., 1998, *Exploring the solution space of semi-structured geographical problems using genetic algorithms*. Transactions in GIS, 3, pp. 51-71.

Bithell, M., Brasington, J. and Richards, K., in press, *Discrete-element, individual-based and agent-based models: Tools for interdisciplinary enquiry in geography?* Geoforum.

Breukers, A., Kettenis, D.L., Mourits, M., Van Der Werf, W. and Oude Lansink, A., 2006, *Individual-based models in the analysis of disease transmission in plant production chains: An application to potato brown rot*. Agricultural Systems, 90, pp. 112-131.

Burrough, P.A., 1998, Dynamic Modelling and Geocomputation. In *Geocomputation: a primer*, P.A. Longley, S.M. Brooks, R. McDonnel and B. MacMillan (Eds.), pp. 165-191 (Chichester: Wiley).

Coley, D.A., 1999, *An introduction to genetic algorithms for scientists and engineers* (Singapore: World Scientific Publishing Co.).

Gimblett, R., Lynch, J., Daniel, T., Ribes, L. and Oye, G., 2003, *Deriving artificial models of visitors from dispersed patterns of use in the Sierra Nevada Wilderness, California*. Journal for Nature Conservation, 11, pp. 287-296.

Giupponi, C., Jakeman, A.J., Karssenberg, D. and Hare, M.P. (Eds.), 2006, *Sustainable management of water resources: an integrated approach* (Cheltenham: Elgar).

Goodchild, M.F., Yuan, M. and Cova, T.J., 2007, *Towards a general theory of geographic representation in GIS*. International Journal of Geographical Information Science, 21, pp. 239-260.

Groff, E.R., 2007, *'Situating' simulation to model human spatio-temporal interactions: An example using crime events*. Transactions in GIS, 11, pp. 507-530.

Helbing, D., Farkas, I. and Vicsek, T., 2000, *Simulating dynamical features of escape panic*. Nature, 407, pp. 487-490.

Janssen, P.H.M. and Heuberger, P.S.C., 1995, *Calibration of process-oriented models*. Ecological Modelling, 83, pp. 55-66.

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

Karssenberg, D. and De Jong, K., 2005a, *Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions*. International Journal of Geographical Information Science, 19, pp. 559-579.

Karssenberg, D. and De Jong, K., 2005b, *Dynamic environmental modelling in GIS: 2. Modelling error propagation*. International Journal of Geographical Information Science, 19, pp. 623-637.

Karssenberg, D. and De Jong, K., 2006a, Towards improved solution schemes for Monte Carlo simulation in environmental modeling languages. In *Geo-information and computational geometry*, P.J.M. Oosterom and M.J. Kreveld (Eds.) (Delft: NCG Nederlandse Commissie voor Geodesie, Netherlands Geodetic Commission).

Karssenberg, D., Törnqvist, T.E. and Bridge, J.S., 2001, *Conditioning a process-based model of sedimentary architecture to well data*. Journal of Sedimentary Research, 71, pp. 868-879.

Karssenberg, D., Pfeffer, K. and Visssers, M., 2006b, Software tools for hydrological modelling. In *Sustainable management of water resources: an integrated approach*, C. Giupponi, A.J. Jakeman, D. Karssenberg and M.P. Hare (Eds.), pp. 235-262 (Cheltenham: Elgar).

Karssenberg, D., De Jong, K. and Van Der Kwast, J., 2007, *Modelling landscape dynamics with Python*. International Journal of Geographical Information Science, 21, pp. 483-495.

Ligtenberg, A., Wachowicz, M., Bregt, A.K., Beulens, A. and Kettenis, D.L., 2004, *A design and application of a multi-agent system for simulation of multi-actor spatial planning*. Journal of Environmental Management, 72, pp. 43-55.

Mclaughlin, D. and Townley, L.R., 1996, *A reassessment of the groundwater inverse problem*. Water Resources Research, 32, pp. 1131-1161.

Moulin, B., Chaker, W. and Gancet, J., 2004, *PADI-Simul: An agent-based geosimulation software supporting the design of geographic spaces*. Computers, Environment and Urban Systems, 28, pp. 387-420.

Pcraster, 2006, PCRaster internet site. Available online at: http://pcraster.geo.uu.nl (accessed Oct 2007 2007).

Pullar, D., 2001, *MapScript: A map algebra programming language incorporating neighborhood analysis*. Geoinformatica, 5, pp. 145-163.

Python, 2007, Python programming language. Available online at: http://www.python.org (accessed 15 oct 2007).

Sydelko, P.J., Hlohowskyj, I., Majerus, K., Christiansen, J. and Dolph, J., 2001, *An object-oriented framework for dynamic ecosystem modeling: Application for integrated risk assessment*. Science of the Total Environment, 274, pp. 271-281.

Tomlin, C.D., 1990, *Geographic Information Systems and Cartographic Modeling* (New York: Prentice Hall).

Van Deursen, W.P.A., 1995, *Geographical Information Systems and Dynamic Models* (Utrecht: Koninklijk Nederlands Aardrijkskundig Genootschap/Faculteit Ruimtelijke Wetenschappen, Universiteit Utrecht).

Van Deursen, W.P.A., Wesseling, C. and Karssenberg, D., 2000, How do we gain control over GIS technology? In *International Conference on Integrating Geographic Information Systems and Environmental Modeling: Problems, Prospectus, and Needs for Research.*B.O. Parks, K.M. Clarke and M.P. Crane (Eds.), Banff, Canada: Boulder: University of Colorado - Cooperative Institute for Research in Environmental Sciences, Denver: US Geologic Survey - Center for Biological Informatics, and Boulder: NOAA National Geophysical Data Center - Ecosystem Informatics).

Van Leeuwen, P.J., 2003, *A variance-minimizing filter for large-scale applications*. Monthly Weather Review, 131, pp. 2071-2084.

Weerts, A.H. and El Serafy, G.Y.H., 2006, *Particle filtering and ensemble Kalman filtering for state updating with hydrological conceptual rainfall-runoff models*. Water Resources Research, 42.

Wesseling, C.G., Karssenberg, D., Van Deursen, W.P.A. and Burrough, P.A., 1996, *Integrating dynamic environmental models in GIS: the development of a Dynamic Modelling language.* Transactions in GIS, 1, pp. 40-48.