Stochastic spatio-temporal modelling with PCRaster Python

Derek Karssenberg, Oliver Schmitz, Kor de Jong, Faculty of Geosciences, Utrecht University, the Netherlands, info@pcraster.eu

Stochastic spatio-temporal modelling

Model

$$\mathbf{z}_t = f(\mathbf{z}_{t-1}, \mathbf{i}_t, \mathbf{p}_t) \quad \text{for all time steps } t = 1, 2, \ldots, T$$

state variables — *transition function* — *inputs* — *parameters*

Solution scheme

for each n in Monte Carlo samples:
    for each t in time steps:
        $\mathbf{z}_t^{(n)} = f(\mathbf{z}_{t-1}^{(n)}, \mathbf{i}_t^{(n)}, \mathbf{p}_t^{(n)})$

Building blocks

```
discharge = kinematic(flowDir,precipitation,..)
```

result map — *spatial function* — *input maps*

Building blocks to construct the transition function are functions on spatial data types (raster maps). Functions were developed in C++ and are available as Python functions (Python extension).

Solution framework (Python)

```
from PCRaster import *
from PCRaster.Framework import *

class SnowModel(DynamicModel, MonteCarloModel):
  def __init__(self):
    ...
  def premcloop(self):
    dem = self.readmap('dem')
    self.ldd = lddcreate(dem, ...)
    ...
  def initial(self):
    self.snow = scalar(0)
    ...
  def dynamic(self):
    runoff = accuflux(self.ldd, rain)
    self.report(runoff, 'q')
    ...
  def postmcloop(self):
    mcpercentiles('q',percentiles,..)
```

sets constant variables and parameters

is run at t = 0 for each Monte Carlo sample

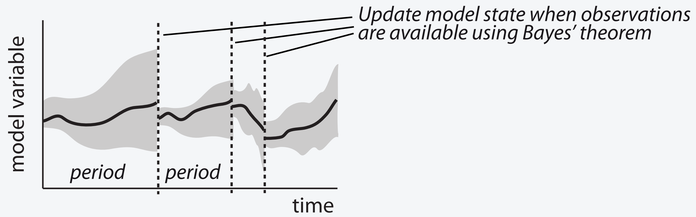
is run for each Monte Carlo sample and for each time step

is run at end calculating sampling statistics over Monte Carlo samples

PCRaster

- Is targeted at the development of spatio-temporal models
- Fast model development and execution
- Scripting environments: PCRcalc and Python
- Rich set of model building blocks for manipulating raster maps
- Framework for stochastic spatio-temporal model building
- Framework for data assimilation
- Tool for visualisation of spatio-temporal stochastic data
- Runs on Linux, Microsoft Windows and Apple OS X.
- Can be downloaded for free!

Data assimilation



Solution scheme

for each period in periods:
    for each n in Monte Carlo Samples:
        for each t in period:
            $\mathbf{z}_t^{(n)} = f(\mathbf{z}_{t-1}^{(n)}, \mathbf{i}_t^{(n)}, \mathbf{p}_t^{(n)})$
    evaluate Bayes' theorem

Solution framework (Python)

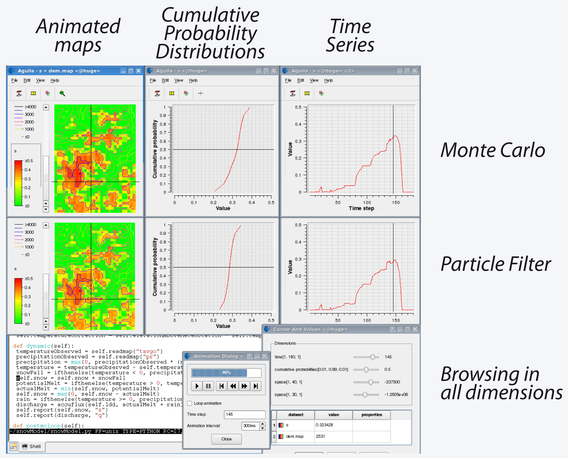
```
def suspend(self):
  self.report(self.snow, 's')
  ...
def updateWeight(self):
  sum = exp(maptotal(((obs - mod)**2)/
        (2.0 * (observedStd ** 2))))
  weight = exp(sum)
  return weight
  ...
def resume(self):
  self.read('s')
  ...
```

store model state at end of period

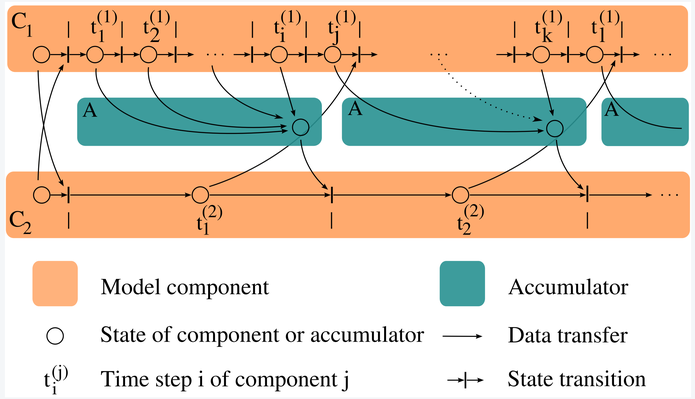
calculate weight of Monte Carlo sample required for solution of Bayes' equation and return to framework

read model state at start of next period

Visualisation of stochastic spatio-temporal data



Current work: integrated modelling



Temporal control flow between model components with shorter (C1) and longer (C2) time steps. Each model component requests output from the other component. C1 directly accepts the input of C2 , C2 expects aggregated values from C1, provided by the accumulator A.

Information and download at:
http://www.pcraster.eu

References
Karssenberg, D., Schmitz, O., Salamon, P., De Jong, K. and Bierkens, M.F.P., 2010, A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. Environmental Modelling & Software, 25, pp. 489-502.