Prepared for:

BAS

Distributed modelling of incoming radiation and precipitation

Distributed hydrological modelling

Research Report

December, 2006

Prepared for:

BAS

Distributed modelling of incoming radiation and precipitation

Distributed hydrological modelling

J. Schellekens

Research Report

December, 2006

Contents

Appendices

1 Introduction

This research describes the work on the project distributed hydrological modelling that was executed in 2006. In previous years the emphasis has been put on developing and extending the actual modelling concepts. In this year the pre-processing of data, specifically evaporation and precipitation that are needed as driving forces to the model(s) will be the focus of the research. In addition, the development of an in-memory interface between pcraster and Delft-Fews has been made operational to allow the pre-processing concepts to be implemented easily.

Evaporation and Precipitation are (obviously) two important parts of a catchment water budget. Unfortunately the determination of the variables at the catchment scale is notoriously difficult. Estimates are hampered by few point measurement or even a complete lack of measurements .

Within a DFID (DFID - UK Department for International Development) funded research project at the VU Amsterdam a distributed catchment model has been developed in PCraster. Two generic parts of that model have be extracted and made available to WL models. This gives us the opportunity to test the newly developed PCraster API that allows Delft-Fews (and other software) to access pcraster functions directly.

With the new functionality now available to Delft-Fews a set of example pcraster modules have been implemented. These are:

- Clearsky/Realsky global radiation, including slope and aspect correction.
- Makkink reference ET for Europe (based on the above).
- Precipitation correction for aspect and slope based on precipitation angle.

2 Samenvatting

Een groot deel van de resultaten van (gedistribueerde) hydrologische modellen worden bepaald door de ruimtelijke invoer gegevens. Met name neerslag en verdamping bepalen de uiteindelijke resultaten. In dit onderzoek zijn twee eerder ontwikkelde modules voor het bepalen van de ruimtelijke verdeling van neerslag en verdamping (gebaseerd op straling) bekeken en met behulp van een operationele koppeling tussen PCraster en Delft-Fews geïmplementeerd. Deze koppeling die in het kader van dit project is geoperationaliseerd maakt het bovendien mogelijk een willekeurig PCraster script op te nemen in Delft-Fews.

De straling en neerslag correctie methodes zijn gebaseerd op eerder onderzoek dat met behulp van ondersteuning van DFID aan de Vrije Universiteit Amsterdam is uitgevoerd.

Aan de hand van deze uitgebreide programma's is een serie simpeler algemeen toepasbare modules ontwikkeld. Deze zijn geïmplementeerd in een demonstratie-versie van Delft-Fews en kunnen gebruikt worden in operationele system, o.a. voor droogte onderzoek:

- Bepaling van Straling voor onbewolkte en bewolkte condities inclusief correctie voor aspect en helling (met behulp van een DEM)
- Bepaling van Makkink verdamping (voorbeeld toepassing voor heel Europa)
- Neerslag correctie met behulp van neerslag hoek en richting en een DEM

3 Status of this research and suggestions for future extensions

Delft-Fews module

At this point a working version is available within Delft-Fews that can be used to perform all operations supported by PCraster. This means that all time series data stored in Delft-Fews (grids and scalars) can be used as input to the module; all output is in grid format. If multiple timsteps are fed to the module at one each timestep will be run separately, i.e. it is not possible to refer to data of a previous timestep within the module. A pcraster model usually consists of a `initial` section (executed only once) and a `dynamic` section that is executed for each timestep. This version only implements the `initial` section.

Next year pcraster environmental software will upgrade the API to include this `dynamic` section. When this is implemented an update to the Delft-Fews module will enable us to also include relatively complicated models within Delft-Fews with relatively little effort. Therefore, it has been planned within next year's research program.

Spatial processing

Within the Fews-GO project AMSR-E microwave data are used to determine top soil water content. This can be combined with potential ET calculation to derive actual Et over large areas. In principle this method has global coverage and relatively high accuracy.

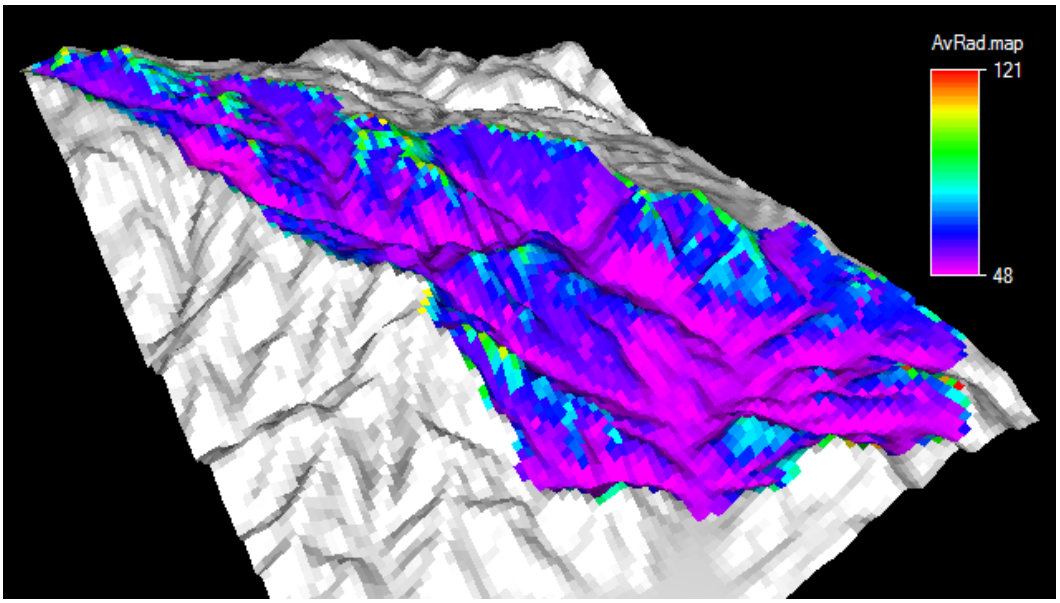
The precipitation correction algorithim cal also be applied to snow-pack buildup. As snow is ligthter the effect will even be larger. In addition, snowmelt modelling is highly dependent on radiation input. It is proposed to setup a pilot project with FOGW are/or the Swiss Federal Institute for Snow and Avalanche Research for a test basin in which the spatial distribution of snow is important. This can be setup as a preprosessing phase for the current (lumped) models which are used in Fews-CH.

4 Distributed (incoming) radiation

4.1 Introduction

In general evaporation amounts are determined for about 90% by radiation input. Radiation at the earth's surface is determined by the potential solar radiation at the edge of the earths atmosphere and the filtering within the atmosphere. The first component can be easily determined from equations. The reduction due to clouds etc can be estimated by incorporation short wave measurements but if these are not available cloud cover estimates can also be used. By combining this with a DEM radiation at the earths surface can be determined including the effects of aspect and shading.

A PCraster base model is available. A sample of spatially distributed radiation output is shown below.



The research will focus on the following points:

1. Filter out any site specific part of the model.
2. Provide options to include or exclude measurement data.
3. Select two evaporation methods that use this model: Penman-Monteith and Makkink.
4. Implement the model using the pcraster API from within Delft-Fews.

4.2 Description[1]

4.2.1 Solar geometry and incoming solar radiation

This section gives a short description. Another description can be found at `http://re.jrc.cec.eu.int/pvgis/pv/solres/solres.htm`.

Potential solar radiation is the radiation of an unobstructed or cloudless sky. The magnitude of the potential solar radiation depends on the position of the sun the solar altitude or solar angle during the day, the inclination of the solar rays with the earth's surface, the amount of radiation at the outer layer of the earth's atmosphere, the transmissivity of the sky and the altitude of the earth's surface.

Solar declination is the annual fluctuation of the sun between the two tropics and varies between –23 and +23 degrees latitude. Solar declination is calculated per *Day* (Julian day number):

(1.1) $$\delta = -23.4 \cos(360 \ (\mathrm{Day} + 10) / 365)$$

The hour angle describes the movement of the sun around the earth in 24 hours, which equals 15 degrees longitude per hour (360o/24h). The hour angle *n* is calculated for each *Hour* (whole hour of the day):

(1.2) $$n = (15 Hour - 12)$$

The position or height of the sun above the horizon is called the solar altitude or solar angle. Solar altitude α (deg) is calculated for each location, determined by the location's latitude φ (deg), declination and hour angle:

(1.3) $$\sin\alpha = \sin\varphi \sin\delta + \cos\varphi \cos\delta \cos n$$

Solar azimuth is the angle between the solar rays and the North-South axis of the earth. Solar azimuth β_s (deg) is calculated by:

(1.4)
$$\cos\beta_s = (\sin\delta \cos\varphi - \cos\delta \sin\varphi \cos n) / \cos\alpha,$$
$$\text{for } Hour \le 12 : \beta_s = \beta_s,$$
$$\text{for } Hour > 12 : \beta_s = 360 - \beta_s$$

Surface azimuth or aspect β_1 (deg) is the orientation of the land surface or slope to the North-South axis of the sun. Slope x (deg) is the maximum rate of change in elevation. Both aspect and slope are predefined functions in PCRaster.

[1] *Taken from van Dam, 2000*

The angle of incidence is the angle between the perpendicular plane of the incoming solar rays and the surface on which they are projected, defined by the aspect and slope of that surface. The angle of incidence ι (deg) is calculated with the solar angle α (deg), the slope of the land surface x (deg), the azimuth of the sun β_s (deg) and azimuth of the land surface β_1 (deg) :

$$\cos\iota = \cos\alpha \sin x \cos(\beta_\sigma - \beta_1) + \sin\alpha \cos x \tag{1.5}$$

The second section of the radiation module calculates the potential solar energy. The amount of solar radiation that reaches the outer atmosphere is decreased by the travelling distance of the solar rays through the sky to the surface, the transmissivity of the sky and the cloud factor.

Solar energy at the outer layer of the atmosphere S_{out} (W.m-2) is calculated by (Kreider & Kreith 1975):

$$S_{out} = S_c(1 + 0.034\cos(360Day/365)) \tag{1.6}$$

where S_c (W.m-2) is the solar constant of 1367 W.m-2 (Duffie & Beckman 1991). The solar 'constant' is subject to much discussion. Gates (1980) gives a value of 1360 W.m-2. The NASA reports a value of 1353 W.m-2 (Jansen 1985), while Duncan *et al.* (1982) give a value of 1367 W.m-2. Monteith and Unsworth (1990) measured the highest value of 1373 W.m-2. The World Radiation Centre uses a value of 1367 W.m-2 (Duffie & Beckman 1991) and this value is also used in this study.

The solar radiation energy that reaches the earth's surface is decreased due to the length of the air mass it has to pass through and the transmissivity τ (% or fraction) of the sky. The radiation flux through a hypothetical plane normal to the beam S_{nor} (W.m-2) is given by (Gates 1980):

$$S_{nor} = S_{out}\tau^{Mh} \tag{1.7}$$

in which M_h (% or fraction) is the relative path length of the optical air mass at altitude h (m). Transmissivity τ is usually between 0.5 and 0.8, but can be as low as 0.4 in the tropics (Whitmore *et al.* 1993), but mostly a value of 0.6 is used (Gates 1980). To calculate the relative path length of an optical air mass at altitude h (m), the relative path length of an optical air mass at sea level M_0 (% or fraction) is corrected for the atmospheric pressure at altitude h. M_h (% or fraction) is calculated using (Kreider & Kreith 1975):

$$M_h = M_0 P_h / P_0 \tag{1.8}$$

in which P_h / P_0 (mbar.mbar-1) is an atmospheric pressure correction. The relative path length of the optical air mass at sea level M_0 is obtained by (Kreider & Kreith 1975):

$$M_0 = \sqrt{(1229 + (614\sin\alpha)^2)} - 614\sin\alpha \tag{1.9}$$

The atmospheric pressure correction P_h / P_0 is written as (List 1984):

$$P_h / P_0 = ((288 - 0.0065h) / 288)^{5.256} \quad (1.10)$$

The incoming radiation normal to the beam S_{nor} must be corrected by the orientation and slope of the surface, defined by the angle of incidence ι, to calculate the incoming radiation S_{dir} (W.m-2) on the earth's surface:

$$S_{dir} = S_{nor} \cos \iota \quad (1.11)$$

Direct light is scattered in the atmosphere. This daylight scattering or diffuse radiation is approximately 15% of direct radiation (Gates 1980). A more accurate empirical estimation for diffuse radiation S_{dif} (W.m-2) in a clear not dust-free sky reads as (Liu and Jordan in Gates 1980):

$$S_{dif} = S_{out}(0.271 - 0.294\tau^{Mh}) \sin \alpha \quad (1.12)$$

During daylight when the sun is above the horizon, it is assumed that all cells receive the same amount of diffuse radiation. Total incoming radiation S_{in} (W.m-2) is the sum of direct and diffuse radiation:

$$S_{in} = S_{dir} + S_{dif} \quad (1.13)$$

Total incoming radiation S_{in} as calculated with equation (1.13) is actually a radiation flux for that moment. In the procedure give above, radiation is calculated per time step, being one hour. If this amount of radiation is used in a water balance model, the amount of radiation and therewith the amount of evapotranspiration will be overestimated or under estimated, depending on the time of the day and the position of the sun.

4.2.2 Shading and correction for aspect

A complete description can be found in the POTRAD manual. Most of the work done for the shading is implemented in the pcraster horizontan function.

4.2.3 Correction for real sky radiation

The methods above calculate clear sky radiation, i.e assuming no cloud cover. To convert this to real sky radiation several methods are commonly used:

1. Use point measurements of radiation to scale the clear sly radiation. *This method is implemented in the example scripts*
2. Use cloud cover data and a regression to convert the real sky radiation. *A standard method for The Netherlands is given in Frantzen en Raaff (1982) If standard cloud cover data is available this can be easily implemented within the current scripts.*
3. Use liquid water path (from space observations) to convert to real sky radiation. *This method has not been tested yet.*

The paragraph below (taken from the r.sun grass manual) describes some of the (other) methods available:

The real-sky irradiance/irradiation are calculated from clear-sky raster maps by the application of a factor parameterizing the attenuation of cloud cover. Examples of explicit calculations of this parameter can be found in Becker (2001), Kitler and Mikler (1986). However, the cloudiness observation by a meteorological service routine is usually prone to subjective errors and does not describe sufficiently the physical nature and dynamic spatial-temporal pattern of different types of cloud cover. Therefore, a simpler parameter has to be used. The solutions for horizontal and inclined surfaces are slightly different. For the assessment of global irradiance/irradiation on a *horizontal surface* under overcast conditions G_h the clear-sky values G_{hc} are multiplied by clear-sky index k_c (Beyer et al 1996, Hammer et al 1998, Rigollier et al. 2001):

$$G_h = G_{hc}\, k_c$$

The index k_c represents the atmospheric transmission expressed as a ratio between horizontal global radiation under overcast and clear-sky conditions. For a set of ground meteorological stations the clear-sky index can be calculated from measured global radiation G_{hs} and computed values of clear-sky global radiation G_{hc}:

$$k_c = G_{hs}/G_{hc}$$

As an alternative the k_c can be derived also from other climatologic data (e.g. cloudiness, cf. Kasten and Czeplak 1980). The raster maps of k_c must be then derived by spatial interpolation. The k_c can be calculated directly as a raster map from short-wave surface irradiance measured by satellites. This method is based on the complementarity between the planetary albedo recorded by the radiometer and the surface radiant flux (Cano et al 1986, Beyer et al 1996, Hammer et al 1998).
To compute the overcast global irradiance/irradiation for *inclined surfaces*, G_i the diffuse D_h and beam B_h components of overcast global radiation and of the clear-sky index k_c have to be treated separately as follows from the equations (26), (27), (29) and (37):

$$D_h = D_{hc}\, k^d_c$$

$$B_h = B_{hc}\, k^b_c$$

The ratio of diffuse to the global radiation D_h/G_h for clear and overcast skies changes according to the cloudiness. In Europe the D_h/G_h values are typically in interval 0.3-1.0 (Kasten and Czeplak 1980). The underlying physical processes are quite complicated and computationally represented only by empirical equations (cf. Scharmer and Greif, 2000, Kasten and Czeplak 1980, Hrvoľ 1991). However, for many meteorological stations, besides the global horizontal radiation G_{hs}, the diffuse component D_{hs} is either measured or calculated from cloudiness, sunshine or other climatologic data. The raster map of D_{hs}/G_{hs} can be derived from the point values by spatial interpolation. Consecutively, the raster maps of diffuse and beam components of the clear sky index can be computed:

$$D_h = G_h\, D_{hs}/G_{hs} \qquad (41)$$

$$B_h = G_h - D_h$$

$$k^d_c = D_h/D_{hc} \quad (42)$$

$$k^b_c = B_h/B_{hc}$$

where subscript *s* is meant to distinguish data measured on meteorological stations B_{hs} and D_{hs} from the estimated values B_h, and D_h.

4.3 Implementation

The method described above has been altered slightly (see the respective sections) and implemented in a number PCraster scripts. These PCraster scripts have been used to create the working examples in DELFT-FEWS describe in the other sections of this document.

4.3.1 DM_InRad

The scripts calculates in coming radiation for all grid cells in a DEM. It needs a number of parameters to determine this. Inputs to the model:

1. A dem in pcraster format;
2. The lat/log coordinates of the centre of the DEM (optionally a Grid file with coordinates for large regions).

Ouputs:

1. Incoming radiation for each cell at hourly timesteps.

4.3.2 DM_PotEvap

This script calculates potential evapotranspiration . Inputs:

1. Outputs from DM_InRad;
2. Surface reflection parameters (Albedo);
3. A method for reducing radiation:
   a) A map/timeseries with cloud cover
   b) A map/timeseries with measured incoming radiation
   c) ???
4. A command line argument to select the potential evap method;
   a) Eo (open water)
   b) PenmanMonth
   c) Makkink
   d) ????
5. A command line options to select the output interval (hourly or daily).

Ouput:

1. Potential evapotranspiration mm/hr of mm/d depending on the method chosen

4.3.3 DM_SnowPack -> Deferred until later

1. Outputs from DM_InRad;
2. A snowpack state file;
3. Precipitation;
4. Temperature -> oops, radiation? _> the swiss snowpack model does all this an (much) better. Should only be done if it take little effort.

4.4 Determining Evapotranspiration

For the purpose of this project a simple evapotransiration module has been made that is based on radiation. Please note that these are **NOT** actual ET estimates as the soil water conditions are not taken into account. They are reference ET estimates based on the available radiation. In this project the Makkink method is used.

4.4.1 Makkink

The Makkink method (Makkink, 1957) can be considered as a simplified Priestley -Taylor formula, requiring, similar to Priestley -Taylor, only radiation and temperature as inputs. The difference is that instead of using net radiation and temperature, the Makkink formula uses incoming short –wave radiation (R_s) and temperature. This can be done because, on average, a constant ratio exists between net radiation (R_n) and short-wave radiation (R_s, ratio ≅ 50%).:

(1.14)
$$E_{ref} = 0.065 \frac{s}{s+\gamma} \frac{R_s}{\lambda}$$

Where:

R_s Downward flux of short-wave radiation W m^{-2} d^{-1}

s Slope of saturated vapour pressure curve kPa $°C^{-1}$

λ Latent heat of evaporation of water W g-1

γ Psychrometer constant kPa °C-1

An advantage of the Makkink formula compared to the Priestley-Taylor formula is that no calculations for long-wave radiation are required. Application of the Makkink formula in Dutch winter months is not possible. This is not necessarily so for other. The above Makkink formula determines reference ET over a grass area. To convert this value to other crops the following table can be used:

Crop factors f as related to Makkink reference - crop evapotranspiration (PET)

	April			May			June			July			August			September		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Grass	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90	0.90	0.90	0.90
Cereals	0.70	0.80	0.90	1.00	1.00	1.00	1.20	1.20	1.20	1.00	0.90	0.80	0.60	-	-	-	-	-
Maize	-	-	-	0.50	0.70	0.80	0.90	1.00	1.20	1.30	1.30	1.20	1.20	1.20	1.20	1.20	1.20	1.20
Potatoes	-	-	-	-	0.70	0.90	1.00	1.20	1.20	1.20	1.10	1.10	1.10	1.10	1.10	0.70	-	-
Sugar beets	-	-	-	0.50	0.50	0.50	0.80	1.00	1.00	1.20	1.10	1.10	1.10	1.20	1.20	1.20	1.10	1.10
Leguminous plants	-	0.50	0.70	0.80	0.90	1.00	1.20	1.20	1.20	1.00	0.80	-	-	-	-	-	-	-
Plant-onions	0.5	0.70	0.70	0.80	0.80	0.90	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-	-	-
Sow-onions	-	0.40	0.50	0.50	0.70	0.70	0.80	0.80	0.90	1.00	1.00	1.00	1.00	0.90	0.90	0.70	-	-
Chicory	-	-	-	-	-	-	0.50	0.50	0.50	0.80	1.00	1.10	1.10	1.10	1.10	1.10	1.10	1.10
Winter carrots	-	-	-	-	-	-	0.50	0.50	0.50	0.80	1.00	1.10	1.10	1.10	1.10	1.10	1.10	1.10
Celery	-	-	-	-	-	0.50	0.70	0.70	0.70	0.80	0.90	1.00	1.10	1.10	1.10	1.10	1.10	-
Leek	-	-	-	-	0.50	0.50	0.50	0.50	0.70	0.70	0.80	0.80	0.80	1.00	0.90	0.90	0.90	0.90
Bulb/tube crops	-	-	-	-	0.50	0.70	0.70	0.90	1.20	1.20	1.20	1.20	1.20	1.20	1.20	1.20	1.20	1.20
Pome/stone-fruit	1.00	1.00	1.00	1.40	1.40	1.40	1.60	1.60	1.60	1.70	1.70	1.70	1.30	1.30	1.20	1.20	1.20	1.20

R. A. Feddes: Crop factors in relation to Makkink reference-crop-evapotranspiration. In: Evaporatranpiration and Weather. TNO Comm. on Hydrol. Res., Proceed. and Inform., no. 39, 1987

Figure 1 Crop factors for converting Eref to Eact (Feddes, 1987)

Some considerations when using the Makkink formula:

- Note that the Makkink formula does not produce reliable results in winter time;
- it has been designed for daily averages;
- The crop factors given assume to water stress occurs.

A more more complete representation of the evaporation process is given by the penman-Monteith formula:

$$(1.15)\ \lambda ET_o = \frac{\Delta (R_n - G) + \rho c_p (e_a - e_d) / r_a}{\Delta + \gamma (1 + r_c / r_a)}$$

where:
- λET_o : latent heat flux of evaporation [kJ m^{-2} s^{-1}]
- R_n : net radiation flux at surface [kJ m^{-2} s^{-1}]
- G : soil heat flux [kJ m^{-2} s^{-1}]
- ρ : atmospheric density [kg m^{-3}]
- c_p : specific heat moist air [kJ kg^{-1} °C^{-1}]
- (e_a-e_d) : vapour pressure deficit [kPa]
- r_c : crop canopy resistance [s m^{-1}]
- r_a : aerodynamic resistance [s m^{-1}]
- Δ : slope vapor pressure curve [kPa °C^{-1}]
- γ : psychrometric constant [kPa °C^{-1}]
- λ : latent heat of vaporization [MJ kg^{-1}]

5 Correction Precipitation for windspeed and landscape aspect and slope

5.1 Introduction[2]

Information about local surface slope and aspect as well as precipitation inclination and direction are needed for the proper estimation of atmospheric water inputs to montane areas of complex topography. The inclination of precipitation (α) is expected to vary as a function of wind speed and turbulence, even over small scales. Furthermore, fog is intercepted differently by tall forest and short pasture vegetation. As such, spatial variability of rainfall, drizzle, and fog deposition in montane areas is high and this needs to be accounted for if reliable estimates of spatial inputs are to be obtained. Similarly, snowpack accumulation is also influenced by these factors.

5.2 Description

The total liquid water in near-surface montane air consists of any rain, drizzle and fog present and these may precipitate onto a surface or stay in suspension depending on droplet size, wind speed and turbulence. The *potential precipitation* (P_{pot}) at a given point may be defined as the potential water depth (mm) reaching a plane that is orientated perpendicularly to the average trajectory direction of the water drops over a specified period of time. The vertical component of P_{pot} is measured as precipitation (P) by a conventional rain gauge.

In this research we use a simple regression between windspeed and P intensity to determine the precipitation angle (and thus the potential precipitation). Such a regression is site specific. Other, more elaborate methods are described in Frumau, 2006 and in the cqflow model documentation (also in the project CD).

The first step is to determine the angle of precipitation. Using this angle we can determine the precipitation in a plane perpendicular o the rainfall trajectory:

$$P_{pot} = P_{gauge} / \cos(\alpha) \tag{1.16}$$

In which P_{gauge} is the amount measure using a normal raingauge and α the angle of the precipitation. Using a DEM the input angle χ of the precipitation relative to the terrain can be determined:

$$\chi = \text{acos}(\cos(\beta)\cos(\alpha) + \sin(\beta)\sin(\alpha)\cos(Asp) - \sin(PAsp)) \tag{1.17}$$

[2] *Adjusted from Frumau et al 2006*

In which β is the Terain angle and *Asp* and *PAsp* are the terrain en precipitation aspect (Wind direction) respectively. Using this we can calculate the precipitation on the sloping surface (P_{slope}):

$$P_{slope} = P_{pot} \cos(\chi) \tag{1.18}$$

To convert this to the amount of P on a the horizontal plane (P_{cor}) we can use:

$$P_{cor} = P_{slope} / \cos(\beta) \tag{1.19}$$

6 Integration into Delft Fews, the pcrtransformationcontroller

6.1 Introduction

Delft-Fews has been modified to be able to use the Pcraster XML api. This api allows in memory transfer of data. To do this a system plugin has been made, the pcrtransformationcontroller.

This plugin can be configured to extract data from the Fews database, used this as input to a pcraster script and retrieved selected results which can be saved into the database fro further processing or display.

6.2 Issues and limitations

- Currently no dynamic section can be used. As such the systems runs the script for each timestep, the systems can only send single grids and single values to pcraster at once. As such some constructions that rely on timestepping will not work.
- Delft-Fews has no concept (yet) of static grid data. As such the DEM has to be read into the systems before each run which may be time-consuming. This is not a limitation of the pcraster module and once the static data concept has been put into the system (next year) the change should be transparent to this module.
- Delft-Fews grids can be dynamic: each timestep can be a different size. Pcraster does not support this. In the current implementation this is not an issues as pcraster is initialized each timestep but this has to be dealt with later.

7 Working Configuration Examples

7.1 Introduction

To demonstrate the module several examples have been set up within Delft-Fews. The system can be started by running the `pcrfews.exe` file in the bin directory.

To see the results (input/output) the system time must be set to 14 Nov 2006 12:00

All results can been displayed using the grid display. The following workflows are available in the system:

Workflow	**required T0**	**Description**
RadiationClearSky	none	This version determines Clear Sky radiation assuming a level surface using a uniform altitude. No shading correction.
RadiationRealSky	First enter measured values for the radiation gauges.	This version estimates Real Sky radiation by first determining using a Kc factor. In this version a horizontal DEM is assumed and the ratio between diffuse and direct (beam) radiation is NOT adjusted. The following methods can be used to determine real sky radiation: 1) supply timeseriesSet of measured global radiation to determine Kc by: Kc = Measured-Radiation/Computed Clear sky.

RadiationClearSkyOnDem	none	This version determines clear sky radiation correct for a DEM. The method assumes the grid dimensions are given in Lat Long and that the Altitude in the DEM is given in meters. The user supplied CelLength in meters is used to scale the DEM for the Slope and critical sun angle calculations.
RadiationClearSkyOnDem	2005	This version determines real sky radiation correct for a DEM. The method assumes the grid dimensions are given in Lat Long and that the Altitude in the DEM is given in meters. The user supplied CelLength in meters is used to scale the DEM for the Slope and critical sun angle calculations. To use the method the ImportExternal workflow must be run and the T0 should be set in 2005.
MakkinkEurope	First enter measured values for the temperature gauges.	
PrecipitationOnDem	First enter measured values for the rain, windspeed and winddir gauges Select Gauges Costa Rica: • enter valuess for measured P, windpseed and direction • Run workflow	

7.2 RadiationClearSky

This script (implemented in the RadiationClearSky moduleInstance) determines clear sky radiation for a grid. This example grid covers most of Europe.

Inputs:

- A time series with hour of day;
- A time series with day of the year.

Outputs:

- A grid with total clear sky solar radiation;
- A grid with direct clear sky solar radiation;
- A grid with diffuse clear sky radiation.

Configuration in Delft-Fews

```
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>Radiation</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="YearDay" dataType="scalar"
convertDatum="false" scalarType="timeAsDayofYear">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Time.event</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="Hour" dataType="scalar"
convertDatum="false" scalarType="timeAsHourofDay">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Time.event</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <!-- Total potential Solar radiation -->
                        <outputVariable variableId="SL" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
```

```
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- Diffuse radiation -->
                        <outputVariable variableId="SLDF" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.diffuse</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- direct radiation -->
                        <outputVariable variableId="SLDR" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.direct</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="String">
            <text><![CDATA[
```

Script code

```
# Test script to determine radiation over a grid.
#
# Inputs from Delft-Fews into this script
# - YearDay -> scalar with  day since beginning of year
# - Hour of day -> Fractional hour of day (e.g. 12.5 = 12:30)
# Ouputs to FEWS
# - SL -> Total Solar radiation
#
#  This version determines Clear Sky radiation assuming a level surface using a
uniform
# altitude. This level is configured in the script below.
Altitude=spatial(0);

Latitude = ycoordinate(boolean(Altitude));
Longitude = xcoordinate(boolean(Altitude));

Day =YearDay;
pi = 3.1416;
Sc       = 1367.0;          # Solar constant (Gates, 1980) [W/m2]
Trans    = 0.6;             # Transmissivity tau (Gates, 1980)

AtmPcor = ((288-0.0065*Altitude)/288)**5.256;            # atm pressure corr [-]

# Solar geometry
# ----------------------------
# SolDec  :declination sun per day  between +23 and -23 [deg]
# HourAng :hour angle [-] of sun during day
# SolAlt  :solar altitude [deg], height of sun above horizon
# SolDec  = -23.4*cos(360*(Day+10)/365);
```

```
# Now added a new function that should work on all latitudes!
theta    =(Day-1)*360/365;  # day expressed in degrees

# Time change equal to 4 min per degree longtitude
# Assume the time input to be GMT
HourS = Hour + (Longitude * 4/60);

SolDec =180/pi * (0.006918-0.399912 * cos(theta)+0.070257 * sin(theta) -   0.006758 *
cos(2*theta)+0.000907   *   sin(2*theta)   -     0.002697   *   cos(3*theta)+0.001480   *
sin(3*theta));

HourAng = 15*(HourS-12.01);

SolAlt  = scalar(asin(scalar(sin(Latitude)*sin(SolDec)+cos(Latitude)*
           cos(SolDec)*cos(HourAng))));

# Solar azimuth
# ----------------------------
# SolAzi  :angle solar beams to N-S axes earth [deg]
 SolAzi = scalar(acos((sin(SolDec)*cos(Latitude)-cos(SolDec)*
          sin(Latitude)*cos(HourAng))/cos(SolAlt)));
 SolAzi = if(HourS le 12 then SolAzi else 360 - SolAzi);

 Slope = 0.0001;
 Aspect = 1;

# Surface azimuth
# ----------------------------
# cosIncident :cosine of angle of incident; angle solar beams to angle surface
 cosIncident = sin(SolAlt)*cos(Slope)+cos(SolAlt)*sin(Slope)
               *cos(SolAzi-Aspect);

# Radiation outer atmosphere
# ----------------------------
 OpCorr = Trans**((sqrt(1229+(614*sin(SolAlt))**2)
          -614*sin(SolAlt))*AtmPcor);    # correction for air masses [-]
 Sout   = Sc*(1+0.034*cos(360*Day/365)); # radiation outer atmosphere [W/m2]
 Snor   = Sout*OpCorr;                  # rad on surface normal to the beam [W/m2]

# Radiation at DEM
# ----------------------------
# Sdir   :direct sunlight on a horizontal surface [W/m2] if no shade
# Sdiff  :diffuse light [W/m2] for shade and no shade
# Stot   :total incomming light Sdir+Sdiff [W/m2] at Hour
# Radiation :avg of Stot(Hour) and Stot(Hour-HourStep)
# NOTE: PradM only valid for HourStep and DayStep = 1
 Sdir   = if(Snor*cosIncident<0,0.0,Snor*cosIncident);
 Sdiff  = if(Sout*(0.271-0.294*OpCorr)*sin(SolAlt)<0, 0.0,
          Sout*(0.271-0.294*OpCorr)*sin(SolAlt));

# Fill in missing values with areaaaverage
SLDR=cover((Sdir*1),(Altitude * 0) + areaaverage(Sdir*1,boolean(Altitude))); # hourly
rad [W/m2]
SLDF=cover((Sdiff*1),(Altitude  *  0)  +  areaaverage(Sdiff*1,boolean(Altitude)));  #
hourly rad [W/m2]

SL   = SLDR + SLDF;       # Total rad  in [W/m2]
```

Example output

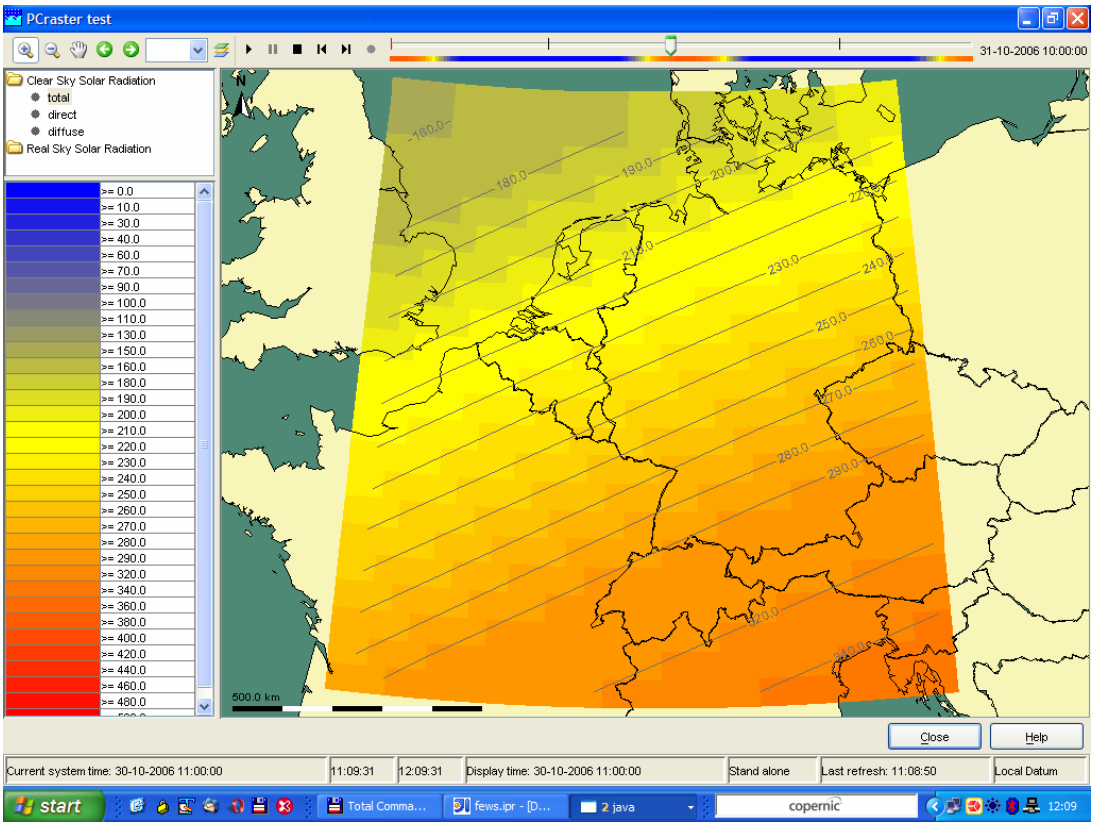


Figure 2 Example showing calculated clear sky radiation for a grid.

7.3 RadiationRealSky

This script (implemented in the RadiationRealSky moduleInstance) determines real sky radiation for a grid. This example grid covers most of Europe.
One or more timeseries with measure radiation are used to correct the potential radiation. Missing values in the measured radiation are allowed. If *all* gauges are missing for a timestep the script returns clear sky radiation.

Inputs:

- A time series with hour of day;
- A time series with day of the year;
- One or more time series of measured incoming radiation. These locations have to fall within the grid.

Outputs:

- A grid with total real sky solar radiation;
- A grid with direct real sky solar radiation;
- A grid with diffuse real sky radiation.

Configuration in Delft-FEWS

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solar radiation module demonstration configuration
       This version estimates Real Sky radiation by first determining using a Kc
factor.
       In this version a horizontal DEM is assumed and the ratio between diffuse and
direct (beam)
       radiation is NOT adjusted.
       The following methods can be used to determine real sky radiation:
          1) supply timeseriesSet of measured global radiation to determine Kc by:
Kc = Measured-Radiation/Computed Clear sky.
          -->
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
       <logLevel>WARN</logLevel>
       <pcrTransformationSet id="Potradiation">
              <areaMap>
                     <locationId>Radiation</locationId>
              </areaMap>
              <definitions>
                     <dataExchange>memory</dataExchange>
                     <inputVariable variableId="YearDay" dataType="scalar"
convertDatum="false" scalarType="timeAsDayofYear">
                            <timeSeriesSet>
                                   <moduleInstanceId>Radiation</moduleInstanceId>
                                   <valueType>scalar</valueType>
                                   <parameterId>Time.event</parameterId>
                                   <locationId>Radiation</locationId>
                                   <timeSeriesType>external
historical</timeSeriesType>
                                   <timeStep unit="minute" multiplier="15"/>
                                   <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                   <readWriteMode>add originals</readWriteMode>
                            </timeSeriesSet>
                     </inputVariable>
                     <!-- Measured radiation data -->
                     <inputVariable variableId="MeasRad" dataType="scalar"
convertDatum="false" spatialType="spatial">
                            <timeSeriesSet>
                                   <moduleInstanceId>Radiation</moduleInstanceId>
                                   <valueType>scalar</valueType>
                                   <parameterId>Sol.obs</parameterId>
                                   <locationSetId>Gauges Sol.obs</locationSetId>
                                   <timeSeriesType>external
historical</timeSeriesType>
                                   <timeStep unit="minute" multiplier="15"/>
                                   <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                   <readWriteMode>add originals</readWriteMode>
                            </timeSeriesSet>
                     </inputVariable>
                     <inputVariable variableId="Hour" dataType="scalar"
convertDatum="false" scalarType="timeAsHourofDay">
                            <timeSeriesSet>
                                   <moduleInstanceId>Radiation</moduleInstanceId>
                                   <valueType>scalar</valueType>
                                   <parameterId>Time.event</parameterId>
                                   <locationId>Radiation</locationId>
                                   <timeSeriesType>external
historical</timeSeriesType>
                                   <timeStep unit="minute" multiplier="15"/>
                                   <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                   <readWriteMode>add originals</readWriteMode>
                            </timeSeriesSet>
                     </inputVariable>
                     <!-- Total potential Solar radiation -->
                     <outputVariable variableId="SL" dataType="scalar"
convertDatum="false">
                            <timeSeriesSet>
```

```
        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- Diffuse radiation -->
                        <outputVariable variableId="SLDF" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.diffuse</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- direct radiation -->
                        <outputVariable variableId="SLDR" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.direct</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <outputVariable variableId="MeasMap" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.obs</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="48"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="String">
                        <text><![CDATA[
#! --unittrue --degrees
# Test script for radiation over a DEM. IN this script no hading is used. The DEM is
only
# used to determine altitide. I may be replaced by a fixed value.
#
# Inputs from Delft-Fews into this script
# - Altitude -> a map should be a real DEM later on.
# - YearDay -> scalar with  day since beginning of year
# - Hour of day -> Fractional hour of day (e.g. 12.5 = 12:30)
```

```
# - MeasRad -> measure radiation at a series of gauges. Used to compute k
# Ouputs to FEWS
# - SL -> Total Solar radiation

Altitude=spatial(10); # Use a horizontal DEM

# Creat unique Id's for input stations
Unq = uniqueid(boolean(MeasRad));
# Now generate polygons and fil those
GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
MeasMap = areaaverage(MeasRad,GaugeArea);

Latitude = ycoordinate(boolean(Altitude));
Longitude = xcoordinate(boolean(Altitude));

Day =YearDay;

pi = 3.1416;
Sc       = 1367.0;          # Solar constant (Gates, 1980) [W/m2]
Trans    = 0.6;             # Transmissivity tau (Gates, 1980)

AtmPcor = ((288-0.0065*Altitude)/288)**5.256;            # atm pressure corr [-]

# dynamic No Dynamic in current implementation of FEWS-PCRASTER link
# Solar geometry
# ----------------------------
# SolDec  :declination sun per day  between +23 and -23 [deg]
# HourAng :hour angle [-] of sun during day
# SolAlt  :solar altitude [deg], height of sun above horizon
# SolDec  = -23.4*cos(360*(Day+10)/365);
# Now added a new function that should work on all latitudes!
theta    =(Day-1)*360/365;  # day expressed in degrees

# Time change equal to 4 min per degree longtitude
# Assume the time input to be GMT
HourS = Hour + (Longitude * 4/60);

SolDec =180/pi * (0.006918-0.399912 * cos(theta)+0.070257 * sin(theta) -   0.006758 *
cos(2*theta)+0.000907 * sin(2*theta) -  0.002697 * cos(3*theta)+0.001480 *
sin(3*theta));

HourAng = 15*(HourS-12.01);

SolAlt  = scalar(asin(scalar(sin(Latitude)*sin(SolDec)+cos(Latitude)*
           cos(SolDec)*cos(HourAng))));

# Solar azimuth
# ----------------------------
# SolAzi  :angle solar beams to N-S axes earth [deg]
 SolAzi = scalar(acos((sin(SolDec)*cos(Latitude)-cos(SolDec)*
          sin(Latitude)*cos(HourAng))/cos(SolAlt)));
 SolAzi = if(HourS le 12 then SolAzi else 360 - SolAzi);

 Slope = 0.000001;
 Aspect = 1;

# Surface azimuth
# ----------------------------
# cosIncident :cosine of angle of incident; angle solar beams to angle surface
 cosIncident = sin(SolAlt)*cos(Slope)+cos(SolAlt)*sin(Slope)
               *cos(SolAzi-Aspect);

# Radiation outer atmosphere
# ----------------------------
 OpCorr = Trans**((sqrt(1229+(614*sin(SolAlt))**2)
          -614*sin(SolAlt))*AtmPcor);    # correction for air masses [-]
 Sout   = Sc*(1+0.034*cos(360*Day/365)); # radiation outer atmosphere [W/m2]
 Snor   = Sout*OpCorr;                  # rad on surface normal to the beam [W/m2]

# Radiation at DEM
# ----------------------------
# Sdir   :direct sunlight on a horizontal surface [W/m2] if no shade
# Sdiff  :diffuse light [W/m2] for shade and no shade
# Stot   :total incomming light Sdir+Sdiff [W/m2] at Hour
# Radiation :avg of Stot(Hour) and Stot(Hour-HourStep)
```

```
# NOTE: PradM only valid for HourStep and DayStep = 1
 Sdir   = if(Snor*cosIncident<0,0.0,Snor*cosIncident);
 Sdiff  = if(Sout*(0.271-0.294*OpCorr)*sin(SolAlt)<0, 0.0,
          Sout*(0.271-0.294*OpCorr)*sin(SolAlt));

# Fill in missing values with areaaaverage
SLDR=cover((Sdir*1),(Altitude * 0) + areaaverage(Sdir*1,boolean(Altitude))); # hourly
rad [W/m2]
SLDF=cover((Sdiff*1),(Altitude * 0) + areaaverage(Sdiff*1,boolean(Altitude))); #
hourly rad [W/m2]
SL   = SLDR + SLDF;       # Total rad  in [W/m2]

# Correct radiation total using the gauge information
# In this case we DO NOT change the diffuse to direct ratio

Kc = if(SL > 0.0 then MeasRad/SL else 0.0); # correction at gauge location
Kc = cover(areaaverage(Kc,GaugeArea),1); # Do average and fill in mssings with one
SLDR = SLDR * Kc;
SLDF = SLDF * Kc;
SL = SLDF + SLDR;
```

```
                    ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

7.4 RadiationClearSkyOnDem

This script (implemented in the RadiationClearSkyOnDem moduleInstance) determines clear sky radiation for a grid using a DEM to correct for aspect and shading. This example grid covers a small area in Costa Rica.

Inputs:

- A time series with hour of day
- A time series with day of the year
- A DEM (implemented using a file link via the pcraster interface).
- A value for the real cell-length in meters

Outputs:

- A grid with total real sky solar radiation
- A grid with direct real sky solar radiation
- A grid with diffuse real sky radiation
- A grid with the DEM

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Solar radiation module demonstration configuration -->
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>RadiationCostaRica</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="CellLength" dataType="scalar"
convertDatum="false">
                                <value>75</value>
                        </inputVariable>
```

```
                        <inputVariable variableId="YearDay" dataType="scalar"
convertDatum="false" scalarType="timeAsDayofYear">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Time.event</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="Altitude" dataType="scalar"
convertDatum="false" spatialType="spatial">

        <external>%REGION HOME%/Modules/Radiation/dem.map</external>
                        </inputVariable>
                        <inputVariable variableId="Hour" dataType="scalar"
convertDatum="false" scalarType="timeAsHourofDay">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Time.event</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <!-- Total potential Solar radiation -->
                        <outputVariable variableId="SL" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- Diffuse radiation -->
                        <outputVariable variableId="SLDF" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.diffuse</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!-- direct radiation -->
                        <outputVariable variableId="SLDR" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot.direct</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
```

```
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <!--Dem, just for fun-->
                        <outputVariable variableId="Dem" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>
                                        <moduleInstanceId>Radiation</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>H.obs</parameterId>
                                        <locationId>RadiationCostaRica</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="0"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="ClearSkyOnDem">
                        <text><![CDATA[
#! --unittrue --degrees
# Test script for radiation over a DEM
#
# Inputs from Delft-Fews into this script
# - Altitude -> a map should be a real DEM later on
# - YearDay -> scalar with  day since beginning of year
# - hour of day
# - Static DEM
# - CellLength in meters
# Ouputs to FEWS
# - SL -> Total Solar radiation
# - SLDR -> Direct solar radiation
# - SLDF -> Diffuse solar radiation
#
# The method assumes the grid dimensions are given in Lat Long and that
# the Altitude in the DEM is given in meters. The user supplied CellLength in meters
# is used to scale the DEM for the Slope and critical sun angle calculations


Latitude = ycoordinate(boolean(Altitude));
Longitude = xcoordinate(boolean(Altitude));


Day =YearDay;

pi = 3.1416;
Sc       = 1367.0;          # Solar constant (Gates, 1980) [W/m2]
Trans    = 0.6;             # Transmissivity tau (Gates, 1980)

AtmPcor = ((288-0.0065*Altitude)/288)**5.256;            # atm pressure corr [-]
ZeroMap=0*scalar(Altitude);                    #map with only zero's

Aspect=scalar(aspect(Altitude));# aspect [deg]
Aspect  = if(Aspect le 0 then scalar(0.001) else Aspect);


Slope=max(0.000001,slope(Altitude*celllength()/CellLength));

# dynamic No Dynamic in current implementation of FEWS-PCRASTER link
# Solar geometry
# ----------------------------
# SolDec  :declination sun per day  between +23 and -23 [deg]
# HourAng :hour angle [-] of sun during day
# SolAlt  :solar altitude [deg], height of sun above horizon
# SolDec  = -23.4*cos(360*(Day+10)/365);
# Now added a new function that should work on all latitudes!
theta    =(Day-1)*360/365;  # day expressed in degrees
```

```
# Time change equal to 4 min per degree longtitude
# Assume the time input to be GMT
HourS = Hour + (Longitude * 4/60);

SolDec =180/pi * (0.006918-0.399912 * cos(theta)+0.070257 * sin(theta) -   0.006758 *
cos(2*theta)+0.000907 * sin(2*theta) -  0.002697 * cos(3*theta)+0.001480 *
sin(3*theta));

HourAng = 15*(HourS-12.01);


SolAlt  = scalar(asin(scalar(sin(Latitude)*sin(SolDec)+cos(Latitude)*
           cos(SolDec)*cos(HourAng))));

# Solar azimuth
# -----------------------------
# SolAzi  :angle solar beams to N-S axes earth [deg]
 SolAzi = scalar(acos((sin(SolDec)*cos(Latitude)-cos(SolDec)*
          sin(Latitude)*cos(HourAng))/cos(SolAlt)));
 SolAzi = if(HourS le 12 then SolAzi else 360 - SolAzi);


# Surface azimuth
# -----------------------------
# cosIncident :cosine of angle of incident; angle solar beams to angle surface
 cosIncident = sin(SolAlt)*cos(Slope)+cos(SolAlt)*sin(Slope)
               *cos(SolAzi-Aspect);


# Critical angle sun
# -----------------------------
# HoriAng  :tan maximum angle over DEM in direction sun, 0 if neg
# CritSun  :tan of maximum angle in direction solar beams
# Shade    :cell in sun 1, in shade 0

 HoriAng   = horizontan(Altitude*celllength()/CellLength,directional(SolAzi));
 HoriAng   = if(HoriAng lt 0 then scalar(0) else HoriAng);
 CritSun   = if(SolAlt gt 90 then scalar(0) else scalar(atan(HoriAng)));
 Shade   = SolAlt gt CritSun;

# Radiation outer atmosphere
# -----------------------------
 OpCorr = Trans**((sqrt(1229+(614*sin(SolAlt))**2)
          -614*sin(SolAlt))*AtmPcor);    # correction for air masses [-]
 Sout   = Sc*(1+0.034*cos(360*Day/365)); # radiation outer atmosphere [W/m2]
 Snor   = Sout*OpCorr;                  # rad on surface normal to the beam [W/m2]

# Radiation at DEM
# -----------------------------
# Sdir   :direct sunlight on a horizontal surface [W/m2] if no shade
# Sdiff  :diffuse light [W/m2] for shade and no shade
# Stot   :total incomming light Sdir+Sdiff [W/m2] at Hour
# Radiation :avg of Stot(Hour) and Stot(Hour-HourStep)
# NOTE: PradM only valid for HourStep and DayStep = 1
 Sdir   = if(Snor*cosIncident<0,0.0,Snor*cosIncident);
 Sdiff  = if(Sout*(0.271-0.294*OpCorr)*sin(SolAlt)<0, 0.0,
          Sout*(0.271-0.294*OpCorr)*sin(SolAlt));

# Apply shading to the direct (beam) component
Sdir = if (Shade then Sdir else 0.0);

# Fill in missing values with areaaaverage
SLDR=cover((Sdir*1),(Altitude * 0) + areaaverage(Sdir*1,boolean(Altitude))); # hourly
rad [W/m2]
SLDF=cover((Sdiff*1),(Altitude * 0) + areaaverage(Sdiff*1,boolean(Altitude))); #
hourly rad [W/m2]

SL   = SLDR + SLDF;       # Total rad  in [W/m2]
Dem = Altitude;                        # Return DEM for display purposes
```

```
                    ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

Example output

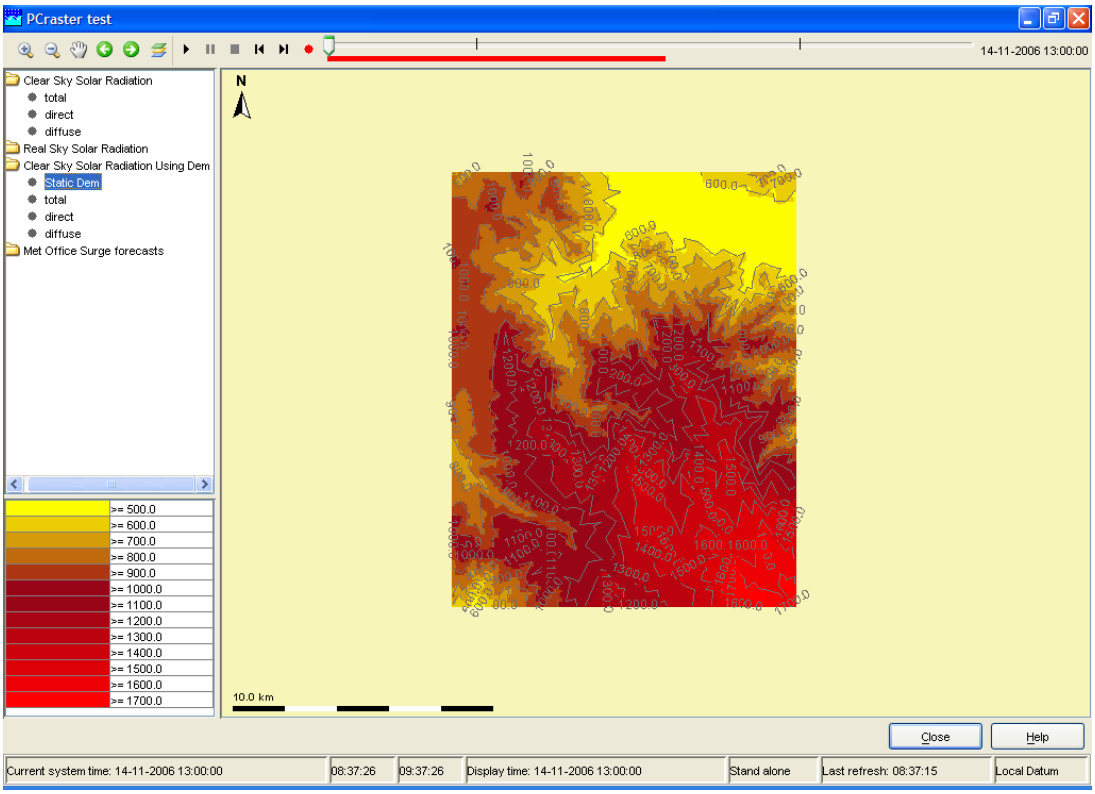


Figure 3 The DEM used in the radiation estimation

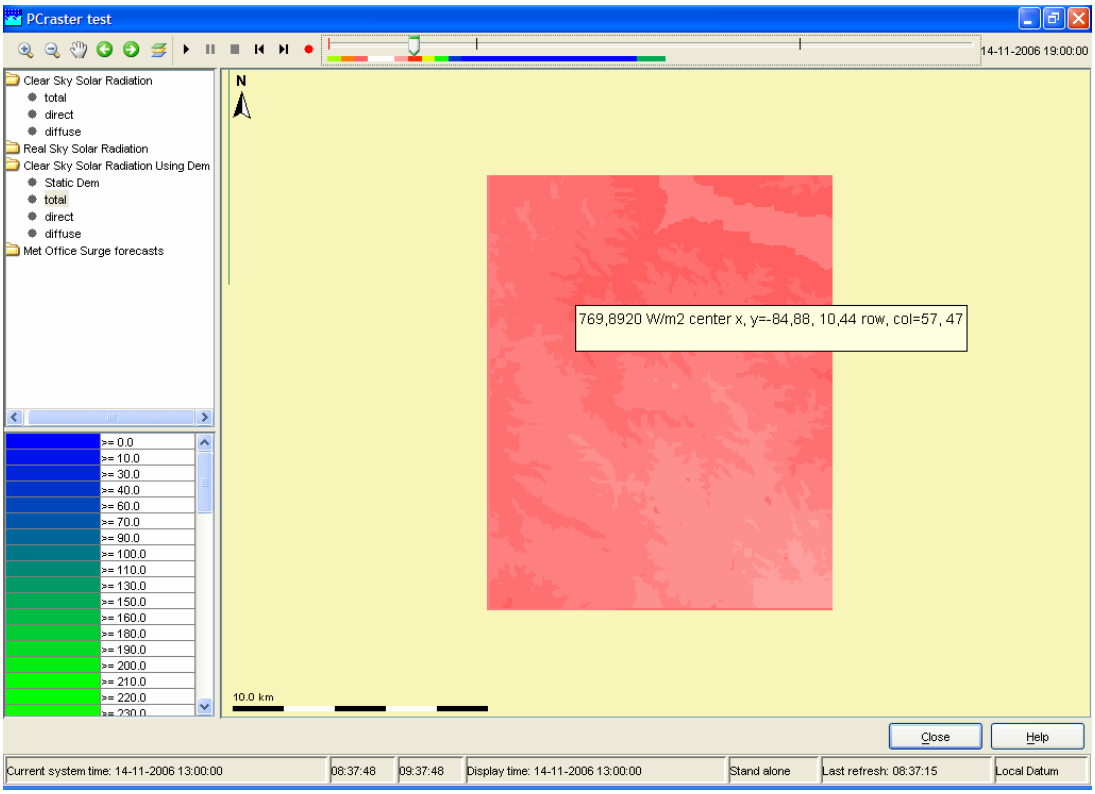


Figure 4 Total clear sky radiation determined for a DEM in Costa Rica

7.5 RadiationRealSkyOnDem

This script (implemented in the RadiationRealSkyOnDem moduleInstance) determines real sky radiation for a grid using a DEM to correct for aspect and slope, including shading. This example grid covers a small part of Costa Rica.
One or more timeseries with measured radiation are used to correct the potential radiation. Missing values in the measured radiation are allowed. If *all* gauges are missing for a timestep the script returns clear sky radiation. This script also tries to determine the correct diffuse to direct radiation fraction on basis of the measure radiation to potential radiation fraction.

Inputs:

- A time series with hour of day;
- A time series with day of the year;
- One or more time series of measured incoming radiation. These locations have to fall within the grid;
- A (static) DEM.

Outputs:

- A grid with total real sky solar radiation;
- A grid with direct real sky solar radiation;
- A grid with diffuse real sky radiation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Makkink module demonstration configuration -->
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>Radiation</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="Rs" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="CropFactor" dataType="scalar"
convertDatum="false" spatialType="spatial">

        <external>%REGION HOME%/Modules/Makkink/cropfactor.map</external>
                        </inputVariable>
                        <inputVariable variableId="Temp" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
```

```
                                        <valueType>scalar</valueType>
                                        <parameterId>T.obs.drybulb</parameterId>
                                        <locationSetId>Gauges_Sol.obs</locationSetId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <!-- Makkink actual Evaporation -->
                        <outputVariable variableId="Eref" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>MakkinkEuropeEact</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>E.makkink.act</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="MakkinkAct">
                        <text><![CDATA[
                        # This scripts determines Makkink reference evaporation
multiplied with a crop factor
                        # after feddes, 1987
                        # inputs:
                        # - Rs -> incoming radiation as a grid
                        # - Temp -> dry bulb temperature (a timeseriesset)
                        # - Cropfactor -> a static map
                        C1 = 0.63;      # Calibration factor, usually between 0.6 and
0.7, may vary with time of year
                        p = 998;        # assume standard pressure
                        cp = 1005;      # standard cp
                        TimestepSecs=900;       # timestep in seconds

                        # Convert the point measurement map to a full map
                        # Creat unique Id's for input stations
                        Unq = uniqueid(boolean(Temp));
                        # Now generate polygons and fil those
                        GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
                        Temp = areaaverage(Temp,GaugeArea);
                        # Use T=15 oC as backup T (and forecast T)
                        Temp = cover(Temp,15);

                        # First determine the slope of saturated vapour pressure curve
(Slope) according to Calder
                        Slope = 10.0 * 2502.9 * exp(17.269 * Temp/(Temp +
237.30))/(Temp + 237.30)**2.0;
                        Tkel = Temp + 273.15;
                        lambda =   4185.5 * (751.78 - (0.5655 * Tkel));
                        Gamma = (cp * p)/(0.622 * lambda);

                        Eref =  CropFactor * C1 * (Slope)/(Slope + Gamma) * Rs; #
W/m^2
                        Eref = Eref/lambda * TimestepSecs; # in mm per timestep

                        ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

Example output

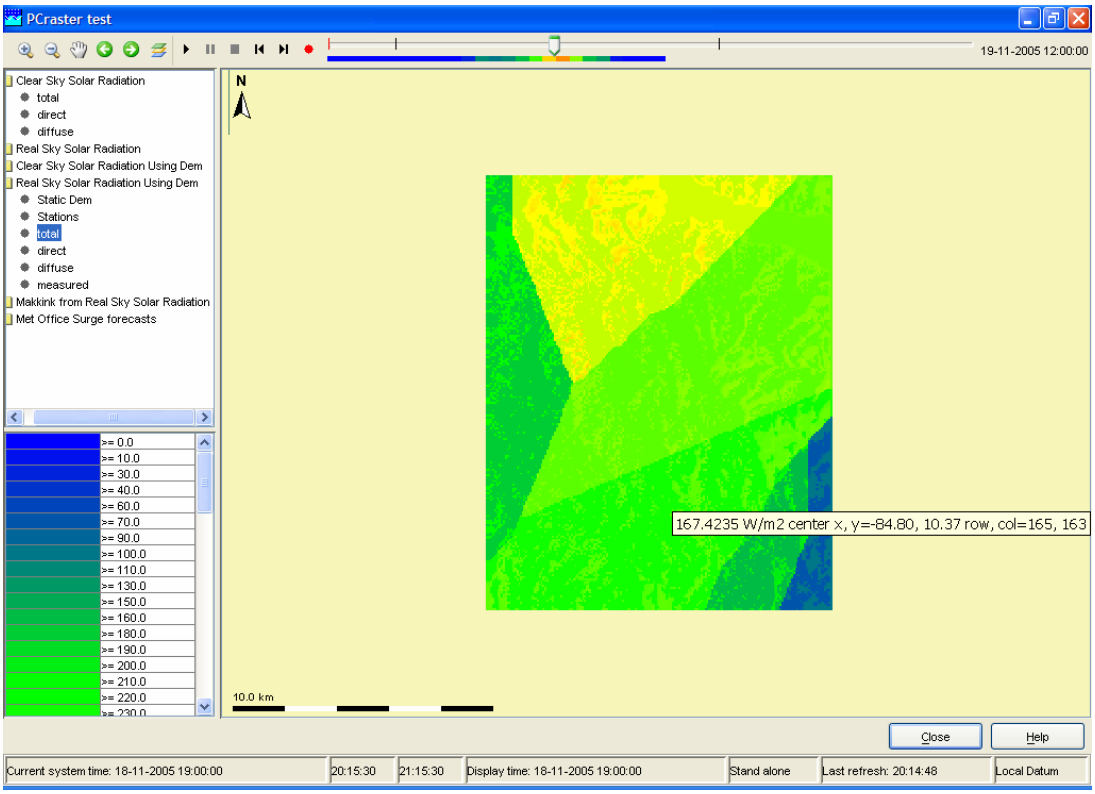


Figure 5 Real sky radiation on a DEM

7.6 MakkinkEuropeEref

The modules determines Makkink reference evaporation (see section 4.4.1) for a grid over Europe. It assumes the RadiationRealSky workflow has already been run. This module is run from the MakkinkEurope workflow.
If the temperature data is missing a default of 15 oC is used. Please also note that the results fro Makkink are reliable only when aggregated to daily totals. This can be done using the transformation utility.

Inputs:

- Incoming radiation (a grid);
- Dry bulb temperature (a set of timeseries).

Output:

- Makkink reference evaporation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Makkink module demonstration configuration -->
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>Radiation</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="Rs" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="Temp" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>T.obs.drybulb</parameterId>
                                        <locationSetId>Gauges Sol.obs</locationSetId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <!-- Makkink reference Evaporation -->
                        <outputVariable variableId="Eref" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>MakkinkEuropeEref</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>E.makkink.ref</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                        <outputVariable variableId="Meas" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>MakkinkEuropeEref</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>T.obs.drybulb</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
```

```
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                    </timeSeriesSet>
                            </outputVariable>
                    </definitions>
                    <pcrModel id="ClearSkyOnDem">
                            <text><![CDATA[
                            # This scripts determines Makkink reference evaporation
                            # inputs:
                            # - Rs -> incoming radiation as a grid
                            # - Temp -> dry bulb temperature (a timeseriesset)
                            C1 = 0.63;      # Calibration factor, usually between 0.6 and
0.7, may vary with time of year
                            p = 998;        # assume standard pressure
                            cp = 1005;      # standard cp
                            TimestepSecs=900;       # timestep in seconds

                            # Convert the point measurement map to a full map
                            # Creat unique Id's for input stations
                            Unq = uniqueid(boolean(Temp));
                            # Now generate polygons and fil those
                            GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
                            Temp = areaaverage(Temp,GaugeArea);
                            # Use T=15 oC as backup T (and forecast T)
                            Temp = cover(Temp,15);

                            # First determine the slope of saturated vapour pressure curve
(Slope) according to Calder
                            Slope = 10.0 * 2502.9 * exp(17.269 * Temp/(Temp +
237.30))/(Temp + 237.30)**2.0;
                            Tkel = Temp + 273.15;
                            lambda =   4185.5 * (751.78 - (0.5655 * Tkel));
                            Gamma = (cp * p)/(0.622 * lambda);

                            Eref =  C1 * (Slope)/(Slope + Gamma) * Rs; # W/m^2
                            Eref = Eref/lambda * TimestepSecs; # in mm per timestep
                            Meas = Temp;

                            ]]></text>
                    </pcrModel>
            </pcrTransformationSet>
</pcrTransformationSets>
```

7.7 MakkinkEuropeEact

The modules determines Makkink ‘actual’ evaporation (see section 4.4.1) for a grid over Europe. It assumes the RadiationRealSky workflow has already been run. The conversion from reference to ‘actual’ is performed using a map with crop factors. This module is run from the MakkinkEurope workflow.
If the temperature data is missing a default of 15 oC is used. Please also note that the results fro Makkink are reliable only when aggregated to daily totals. This can be done using the transformation utility.

Inputs:

- Incoming radiation (a grid);
- Dry bulb temperature (a set of timeseries);
- A static map with crop factors.

Output:

- Makkink ‘actual’ evaporation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Makkink module demonstration configuration -->
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>Radiation</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="Rs" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>RadiationRealSky</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>Sol.pot</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="CropFactor" dataType="scalar"
convertDatum="false" spatialType="spatial">

        <external>%REGION HOME%/Modules/Makkink/cropfactor.map</external>
                        </inputVariable>
                        <inputVariable variableId="Temp" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>T.obs.drybulb</parameterId>
                                        <locationSetId>Gauges Sol.obs</locationSetId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <!-- Makkink actual Evaporation -->
                        <outputVariable variableId="Eref" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>MakkinkEuropeEact</moduleInstanceId>
                                        <valueType>grid</valueType>
                                        <parameterId>E.makkink.act</parameterId>
                                        <locationId>Radiation</locationId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="15"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="MakkinkAct">
                        <text><![CDATA[
                        # This scripts determines Makkink reference evaporation
multiplied with a crop factor
                        # after feddes, 1987
```

```
                        # inputs:
                        # - Rs -> incoming radiation as a grid
                        # - Temp -> dry bulb temperature (a timeseriesset)
                        # - Cropfactor -> a static map
                        C1 = 0.63;     # Calibration factor, usually between 0.6 and
0.7, may vary with time of year
                        p = 998;       # assume standard pressure
                        cp = 1005;     # standard cp
                        TimestepSecs=900;      # timestep in seconds

                        # Convert the point measurement map to a full map
                        # Creat unique Id's for input stations
                        Unq = uniqueid(boolean(Temp));
                        # Now generate polygons and fil those
                        GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
                        Temp = areaaverage(Temp,GaugeArea);
                        # Use T=15 oC as backup T (and forecast T)
                        Temp = cover(Temp,15);

                        # First determine the slope of saturated vapour pressure curve
(Slope) according to Calder
                        Slope = 10.0 * 2502.9 * exp(17.269 * Temp/(Temp +
237.30))/(Temp + 237.30)**2.0;
                        Tkel = Temp + 273.15;
                        lambda =   4185.5 * (751.78 - (0.5655 * Tkel));
                        Gamma = (cp * p)/(0.622 * lambda);

                        Eref =  CropFactor * C1 * (Slope)/(Slope + Gamma) * Rs; #
W/m^2
                        Eref = Eref/lambda * TimestepSecs; # in mm per timestep

                        ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

Example output

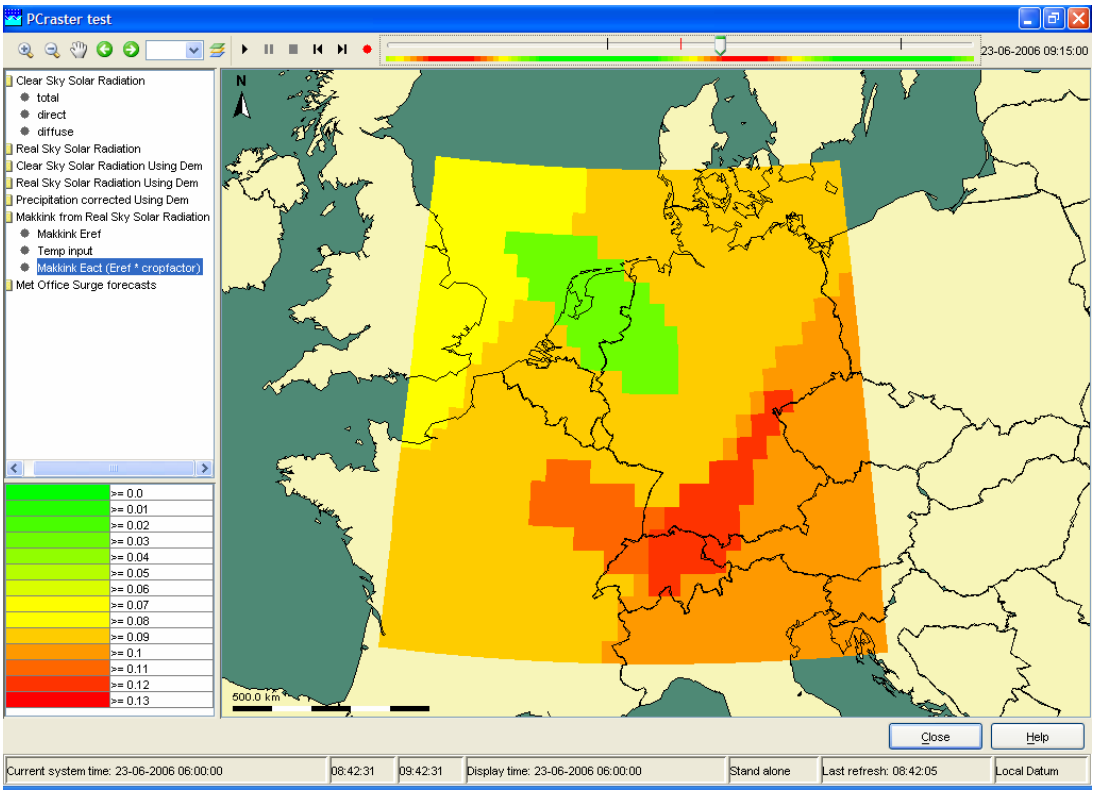


Figure 6 Makking 'actual' ET over europe. The green and red blobs indicate areas for which a different crop factor has been defined

7.8 PrecipitationOnDem

Input:

- P at gauges;
- Windspeed;
- Winddirection.

Output:

- DEM corrected P.

```
<?xml version="1.0" encoding="UTF-8"?>
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews
http://fews.wldelft.nl/schemas/version1.0/pcrTransformationSets.xsd" version="1.1">
        <logLevel>WARN</logLevel>
        <pcrTransformationSet id="Potradiation">
                <areaMap>
                        <locationId>RadiationCostaRica</locationId>
                </areaMap>
                <definitions>
                        <dataExchange>memory</dataExchange>
                        <inputVariable variableId="CellLength" dataType="scalar"
convertDatum="false">
                                <value>75</value>
                        </inputVariable>
                        <!-- Measured precipitation data -->
                        <inputVariable variableId="MeasP" dataType="scalar"
convertDatum="false" spatialType="spatial">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>P.obs</parameterId>
                                        <locationSetId>CRGauges_Sol.obs</locationSetId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="MeasSpeed" dataType="scalar"
convertDatum="false" spatialType="spatial">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Wind.obs.speed</parameterId>
                                        <locationSetId>CRGauges Sol.obs</locationSetId>
                                        <timeSeriesType>external
historical</timeSeriesType>
                                        <timeStep unit="minute" multiplier="60"/>
                                        <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </inputVariable>
                        <inputVariable variableId="MeasDir" dataType="scalar"
convertDatum="false" spatialType="spatial">
                                <timeSeriesSet>

        <moduleInstanceId>ImportTelemetry</moduleInstanceId>
                                        <valueType>scalar</valueType>
                                        <parameterId>Wind.obs.dir</parameterId>
                                        <locationSetId>CRGauges_Sol.obs</locationSetId>
```

```
                                    <timeSeriesType>external
historical</timeSeriesType>
                                    <timeStep unit="minute" multiplier="60"/>
                                    <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                    <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                            </inputVariable>
                            <inputVariable variableId="Altitude" dataType="scalar"
convertDatum="false" spatialType="spatial">

        <external>%REGION HOME%/Modules/Radiation/dem.map</external>
                            </inputVariable>
                            <!-- Corrected Precipitation -->
                            <outputVariable variableId="CorPrecipitation"
dataType="scalar" convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>PrecipitationOnDem</moduleInstanceId>
                                    <valueType>grid</valueType>
                                    <parameterId>P.obs</parameterId>
                                    <locationId>RadiationCostaRica</locationId>
                                    <timeSeriesType>external
historical</timeSeriesType>
                                    <timeStep unit="minute" multiplier="60"/>
                                    <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                    <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                            </outputVariable>
                            <outputVariable variableId="AvgOrgPrec" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>PrecipitationOnDemOrg</moduleInstanceId>
                                    <valueType>grid</valueType>
                                    <parameterId>P.obs</parameterId>
                                    <locationId>RadiationCostaRica</locationId>
                                    <timeSeriesType>external
historical</timeSeriesType>
                                    <timeStep unit="minute" multiplier="60"/>
                                    <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                    <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                            </outputVariable>
                            <outputVariable variableId="AvgCorPrec" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>PrecipitationOnDemCor</moduleInstanceId>
                                    <valueType>grid</valueType>
                                    <parameterId>P.obs</parameterId>
                                    <locationId>RadiationCostaRica</locationId>
                                    <timeSeriesType>external
historical</timeSeriesType>
                                    <timeStep unit="minute" multiplier="60"/>
                                    <relativeViewPeriod unit="hour" start="-24"
end="24"/>
                                    <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                            </outputVariable>
                            <outputVariable variableId="Dem" dataType="scalar"
convertDatum="false">
                                <timeSeriesSet>

        <moduleInstanceId>PrecipitationOnDem</moduleInstanceId>
                                    <valueType>grid</valueType>
                                    <parameterId>H.obs</parameterId>
                                    <locationId>RadiationCostaRica</locationId>
                                    <timeSeriesType>external
historical</timeSeriesType>
                                    <timeStep unit="minute" multiplier="60"/>
                                    <relativeViewPeriod unit="day" start="0"
end="2"/>
```

```
                                        <readWriteMode>add originals</readWriteMode>
                                </timeSeriesSet>
                        </outputVariable>
                </definitions>
                <pcrModel id="CorrectPrecipitation">
                        <text><![CDATA[
#! --unittrue --degrees
# Determines DEM corrected precipitation
# inputs:
# - Static DEM
# - Real cell size in meter
# - Observed Precipitation (locationset)
# - Observed windspeed
# - Observed winddirection
#
# The script used zero precipitation and a zero windspeed as a backup

# NB If for example windspeed fom a NWP grid will be used
# than the lines creating the spatial average can be removed
# and the grid can be used directly. The same applies
# to precipitation and winddirection.
e = 2.7183;

# Determine real slope and the terrain angle.
Slope=max(0.000001,slope(Altitude*celllength()/CellLength));
Terrain_angle=scalar(atan(Slope));

Aspect=scalar(aspect(Altitude));# aspect [deg]
# On Flat areas the Aspect function failes, fill in with average...
Aspect = if (defined(Aspect) then Aspect else areaaverage(Aspect,boolean(Altitude)));

# Create spatial average rain from gauge information.
# Creat unique Id's for input stations
Unq = uniqueid(boolean(MeasP));
# Now generate polygons and fil those
GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
Precipitation = areaaverage(MeasP,GaugeArea);
Precipitation = cover(Precipitation, 0.0);

# Create spatial average Wind from gauge information.
# Creat unique Id's for input stations
Unq = uniqueid(boolean(MeasSpeed));
# Now generate polygons and fil those
GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
WindSpeed = areaaverage(MeasSpeed,GaugeArea);
WindSpeed = cover(WindSpeed, 0.0);

# Create spatial average WindDirection from gauge information.
# Creat unique Id's for input stations
Unq = uniqueid(boolean(MeasDir));
# Now generate polygons and fil those
GaugeArea = spreadzone(ordinal(cover(Unq,0)),0,1);
WindDir = areaaverage(MeasDir,GaugeArea);
WindDir = cover(WindDir, 0.0);

# Determine angle (vertical) of precipitation from windspeed and
# rainfall intensity. NB This is a site specific relation that depends
# on drop size and the drop size to rainfall intensity relation.
b = ln(2+0.38 * Precipitation ** 0.42);
a = 1/ln(e + 10.46 * Precipitation ** 1.24);
PrecipAngle=90- exp(4.5 - a * WindSpeed ** b);

# With the angle and the vertically measured precipitation determine the
# precipitation on a plane at a 90deg angle to the precipitation vector. This
# is the assumed potential real precipitation. This is assumed to be identical for
# each cell in the area.
PotPrecipitation=Precipitation/scalar(cos(PrecipAngle));

# Now use winddirection and angle in combination with the aspect to the angle
# between slope and precipitation. This results in a corrected input in mm.
# The first step is to determine the angle of the terrain from the slope. This has
# been done in the initial section)
# Horizontal precipitation angle determined by the winddir...
PrecipAspect=scalar(WindDir);
```

```
# Now calculate the input angle according to:
Tmp = max(-1,min(1,cos(Terrain_angle)*cos(PrecipAngle)+sin(Terrain_angle)*
sin(PrecipAngle)*cos(scalar(Aspect)-scalar(PrecipAspect))));
Input_angle=scalar(acos(Tmp));

# Remove angles > 90 deg (precipitation going through the ground....)
Input_angle = max (0, min(90, Input_angle));

# now we can calculate the amount of rain on the sloping surface...
CorPrecipitation=PotPrecipitation*scalar(cos(Input_angle));

# Because all the rest of the fluxes are calculated horizontally convert
# the amount back to a horizontal surface...
# this should be equal to precipitation if the PrecipAngle=0
CorPrecipitation=CorPrecipitation/scalar(cos(Terrain_angle));

AvgCorPrec = areaaverage(CorPrecipitation,boolean(Altitude));
AvgOrgPrec = areaaverage(Precipitation,boolean(Altitude));

Dem = Altitude;
                        ]]></text>
                </pcrModel>
        </pcrTransformationSet>
</pcrTransformationSets>
```

8 Products of this research project

8.1 Meetings and Conferences

1. The IAHS Mini-symposium on "Current Challenges and Future of Hydrology" was attended by J. Schellekens on June 28.
2. A presentation was given at the opening of the opgewekt.nu exposition to inform people about the possible risk of flooding. The exposition was opened by secretary of state van Geel. See `opgewekt.nu` and `new-energey.nl` for more information. The lecture programme is given in Appendix B.

8.2 Software

1. Pcraster scripts implementation of the solar radiation module.
2. Pcraster script to perform Precipitation correction.
3. Configurable in memory link between Delft-Fews and the pcraster dll.
4. Update to the pcraster DLL to implement new functionality: a lddcreate that does not include diagonals.
5. Example setup of Delft-Fews that uses the pcraster interface
   a) Radiation examples
   b) Precipitation examples
   c) Example of surge correction suing PCraster

8.3 Papers, reports and presentations

1. This report.
2. The technical design report of the in-memory pcraster link.
3. A ppt presentation describing the research (lecture/lunchlezing to be held next year).

9 References

Frantzen, A.J. en W.R. Raaff, 1982: De relatie tussen de globale straling en de relatieve zonneschijnduur in Nederland. Wetenschappelijk rapport W.R. 82-5,
KNMI, De Bilt.

K.F.Arnoud Frumau, L.A. (Sampurno) Bruijnzeel and Conrado Tobón 2006: Measurement of precipitation in montane tropical catchments: Comparative performance of conventional, spherical and 'potential' rain gauges.

A Sketch Design of the pcraster module for DELFT-FEWS

123

To :
From : J. Schellekens
Subject : pcraster API transformation
Date : 4 July, 2006
Cc :
Action:

Introduction

This memo describes the functional design of a pcrTransformation module. This module will use the new PCRastermemory API to performs transformations on grids (and optional) scalars using the pcraster engine.

Why:
- We can use the PCraster API to perform many operations on grids instead of writing our own. See the end of this document for a list a pcraster functions:
- We can use the pcraster API to do scalar to grid mappsig etc
- The current pcraster adapter cannot handle timeseries as input to the pcraster model (this is required fro many models)

Structure of a pcraster script

For those who have limited experience with pcraster a simple script is shown here:

```
#! --unittrue --degrees

# A comment line

binding # The binding section links variables to maps on the filesystem

Altitude=input\dem.map;                    # DEM


areamap  # The areamap is used as a temple when creating new maps. It is required.
Altitude;

Timer # The timer section determines the number of timesteps 24 in this case
1 24 1;                                    # Day totals

initial # The initial section is used for non dynamic operations on maps and scalars

Latitude  = 5;
Longitude =  5;
```

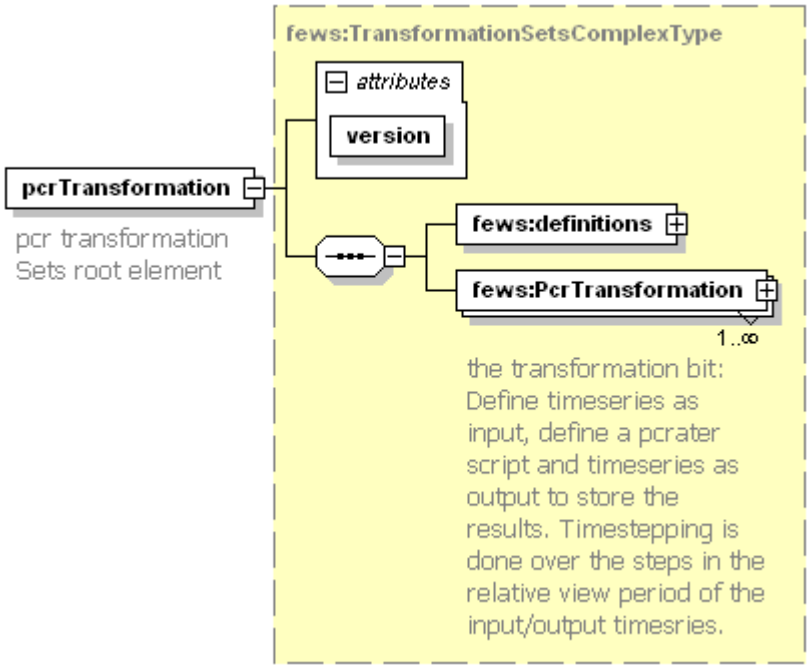
```
CellLength = 23;
Latitude = if (Latitude == 999 then ycoordinate(boolean(Altitude)) else Latitude);
pi = 3.1416;

dynamic # In the dynamic section everthing is done for each timestep This usually contains
the
        # actual model
theta   =(Day-1)*360/365;  # day expressed in degrees
```

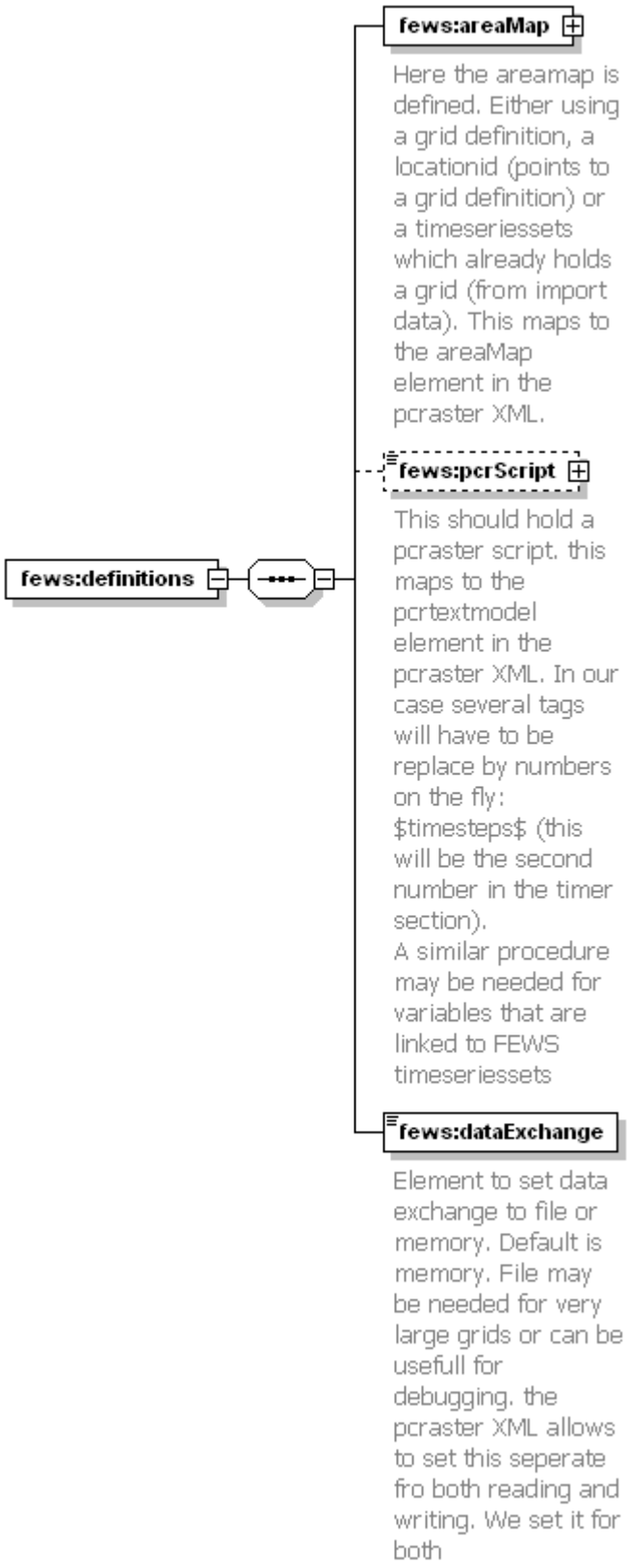
Scope and functionality

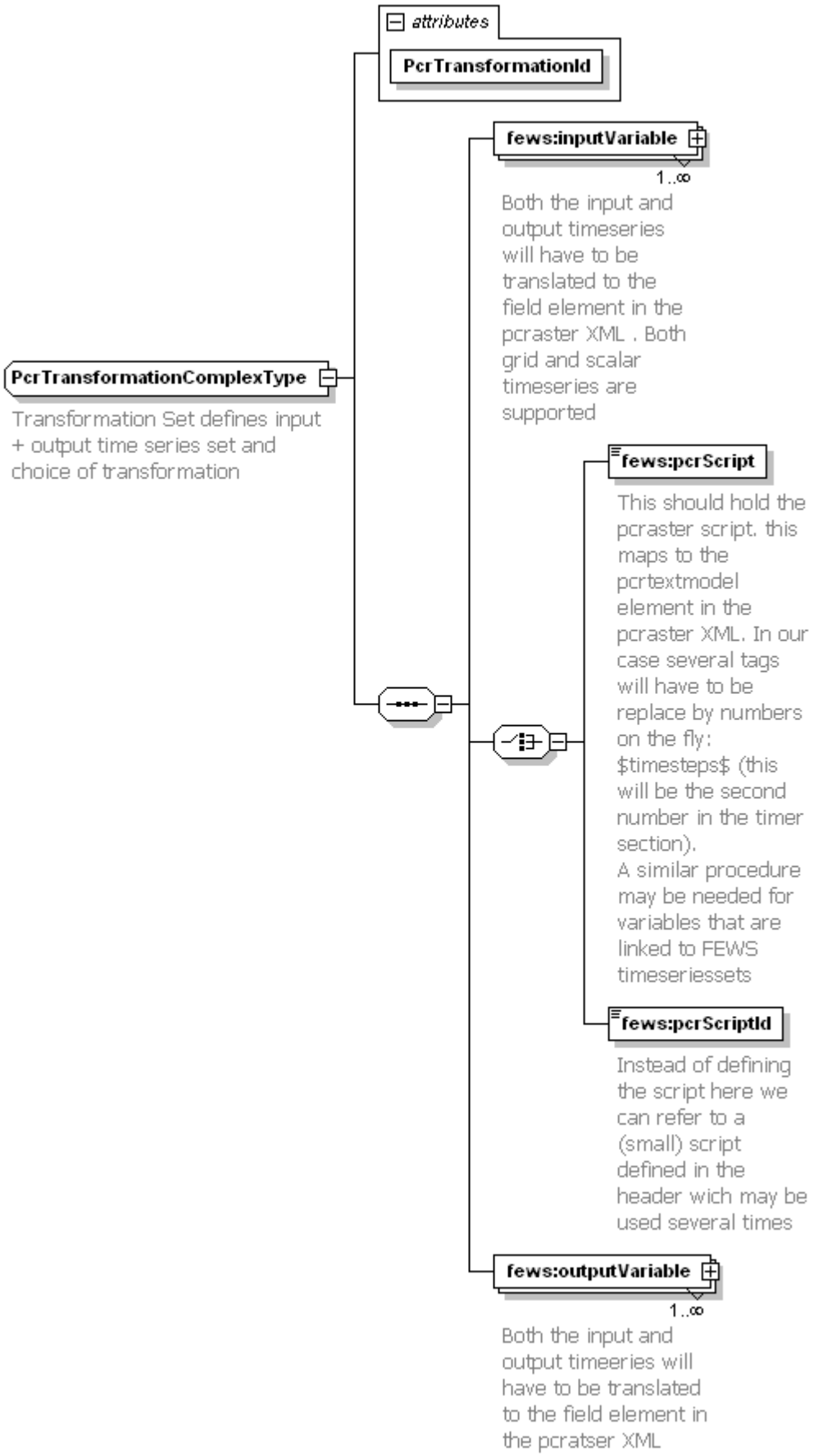
- The module will be able to read multiple input timeseriessets (both grid and scalar) and assign these to a variable (analog to the transformation utility).
- The actual transformation will be a set of pcraster commands (functions) that will use these grids and scalars.
- The module will be able to write multiple output timeseriessets (both grid and scalar) that are linked to variables in the pcraster script.
- The length of the input and output timeseriessets (nr of timesteps) will have to match.
- The module will loop over all the timesteps in the inputtimeseriesset.
- Header.
  - Defines areamap(s). (need to find out how/if we store the (static) areamap). Can this be in the grid definition? I must a a pcraster type map I presume
  - Maybe define scriptlets in the header that can be referred to in transformationsets

Sketch layout of the XML configuration

A xml schema has been made that should be used to configure the module. The schema has two sections: a Headers section (definition of area map etc) and one or PcrTransformations




fews:areaMap
Here the areamap is defined. Either using a grid definition, a locationid (points to a grid definition) or a timeseriessets which already holds a grid (from import data). This maps to the areaMap element in the pcraster XML.
fews:pcrScript
This should hold a pcraster script. this maps to the pcrtextmodel element in the pcraster XML. In our case several tags will have to be replace by numbers on the fly: $timesteps$ (this will be the second number in the timer section).
A similar procedure may be needed for variables that are linked to FEWS timeseriessets
fews:definitions
fews:dataExchange
Element to set data exchange to file or memory. Default is memory. File may be needed for very large grids or can be usefull for debugging. the pcraster XML allows to set this seperate fro both reading and writing. We set it for both


attributes
PcrTransformationId
fews:inputVariable
1..∞
Both the input and output timeseries will have to be translated to the field element in the pcraster XML . Both grid and scalar timeseries are supported
PcrTransformationComplexType
Transformation Set defines input + output time series set and choice of transformation
fews:pcrScript
This should hold the pcraster script. this maps to the pcrtextmodel element in the pcraster XML. In our case several tags will have to be replace by numbers on the fly: $timesteps$ (this will be the second number in the timer section).
A similar procedure may be needed for variables that are linked to FEWS timeseriessets
fews:pcrScriptId
Instead of defining the script here we can refer to a (small) script defined in the header wich may be used several times
fews:outputVariable
1..∞
Both the input and output timeeries will have to be translated to the field element in the pcratser XML

Below is a simple example XML document that uses this schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2006 sp1 U (http://www.altova.com)-->
<pcrTransformation xmlns="http://www.wldelft.nl/fews" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.wldelft.nl/fews:\jaap-
data\Projects\q3947\pcrmodulescehma\pcrTransformation.xsd" version="1.1">
        <definitions>
                <areaMap>
                        <timeSeriesSet>
                                <moduleInstanceId>TWAM_Dataprep</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Wind.u</parameterId>
                                <locationId>MOWind</locationId>
                                <timeSeriesType>simulated forecasting</timeSeriesType>
                                <timeStep unit="minute" multiplier="60"/>
                                <relativeViewPeriod unit="hour" start="-12" end="36"/>
                                <readWriteMode>read only</readWriteMode>
                        </timeSeriesSet>
                </areaMap>
                <dataExchange>memory</dataExchange>
        </definitions>
        <PcrTransformation PcrTransformationId="TestingPCR">
                <inputVariable variableId="inmap" dataType="scalar">
                        <timeSeriesSet>
                                <moduleInstanceId>TWAM_Dataprep</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Wind.u</parameterId>
                                <locationId>MOWind</locationId>
                                <timeSeriesType>simulated forecasting</timeSeriesType>
                                <timeStep unit="minute" multiplier="60"/>
                                <relativeViewPeriod unit="hour" start="-12" end="36"/>
                                <readWriteMode>read only</readWriteMode>
                        </timeSeriesSet>
                </inputVariable>
                <pcrScript><![CDATA[
# it is not clear if we need the timer section. I cannot find a
# reference to it in the pcraster XML as such I have
# assumed we need it here. NB These are pcrater
# comment lines

$timesteps$ 1
= 3.1416;

=inmap * pi;]]></pcrScript>
                <outputVariable variableId="outmap" dataType="scalar">
                        <timeSeriesSet>
                                <moduleInstanceId>examplefile</moduleInstanceId>
                                <valueType>grid</valueType>
                                <parameterId>Wind.u</parameterId>
                                <locationId>MOWind</locationId>
                                <timeSeriesType>simulated forecasting</timeSeriesType>
                                <timeStep unit="minute" multiplier="60"/>
                                <relativeViewPeriod unit="hour" start="-12" end="36"/>
                                <readWriteMode>read only</readWriteMode>
                        </timeSeriesSet>
                </outputVariable>
        </PcrTransformation>
</pcrTransformation>
```

9.1.2 List of pcraster functions

Table of Contents

+ -- Addition
- -- Subtraction
/ or div -- Division
* -- Multiplication
** -- nth power of a first expression, where n is the value of a second expression
abs -- Absolute value
accucapacityflux, accucapacitystate -- Transport of material downstream over a local drain direction network
accuflux -- Accumulated material flowing into downstream cell
accufractionflux, accufractionstate -- Fractional material transport downstream over local drain direction network
accuthresholdflux, accuthresholdstate -- Input of material downstream over a local drain direction network when transport threshold is exceeded
accutriggerflux, accutriggerstate -- Input of material downstream over a local drain direction network when transport trigger is exceeded
acos -- Inverse cosine
and -- Boolean-AND operation
areaarea -- The area of the area to which a cell belongs
areaaverage -- Average cell value of within an area
areadiversity -- Number of unique cell values within an area
areamajority -- Most often occurring cell value within an area
areamaximum -- Maximum cell value within an area
areaminimum -- Minimum cell value within an area
areanormal -- Value assigned to an area taken from a normal distribution
areatotal -- Sum of cell values within an area
areauniform -- Value assignseds to area taken from an uniform distribution
asin -- Inverse sine
aspect -- Aspects of a map using a digital elevation model
atan -- Inverse tangent
boolean -- Conversion data type to boolean data type
catchment -- Catchment(s) of one or more specified cells
catchmenttotal -- Total catchment for the entire upstream area
cellarea -- Area of one cell
celllength -- Horizontal and vertical length of a cell
clump -- Contiguous groups of cells with the same value ('clumps')
cos -- Cosine
cover -- Missing values substituted for values from one or more expression(s)
defined -- Boolean TRUE for non missing values and FALSE for missing values
directional -- Data conversion to the directional data type
downstream -- Cell gets value of the neighbouring downstream cell
downstreamdist -- Distance to the first cell downstream
eq or == -- Relational-equal-to operation on two expressions
exp -- Basee exponential
fac -- Faculty or factorial of a natural positive number
ge or >= -- Relational-greater-than-or-equal-to operation
gt or > -- Relational-greater-than operation
idiv -- Quotient of integer division of values on first expression by values on second expression

if then -- Boolean condition determining whether value of expression or missing value is assigned to result
if then else -- Boolean condition determining whether value of the first or second expression is assigned to result
kinematic -- Dynamic calculation of streamflow through a channel
ldd -- Data conversion from specific data types to local drain direction data type
lddcreate -- Local drain direction map with flow directions from each cell to its steepest downslope neighbour
lddcreatedem -- Modified digital elevation model
ldddist -- Friction-distance from the cell under consideration to downstream nearest TRUE cell
lddmask -- Local drain direction map cut into a (smaller) sound local drain direction map
lddrepair -- Reparation of unsound local drain direction map
le or <= -- Relational-less-than-or-equal-to operation
ln -- Natural logarithm (e)
log10 -- Log 10
lookup -- Compares cell value(s) of one or more expression(s) with the search key in a table
lt or < -- Relational-less-than operation
maparea -- Total map area
mapmaximum -- Maximum cell value
mapminimum -- Minimum cell value
mapnormal -- Cells get non spatial value taken from a normal distribution
maptotal -- Sum of all cell values
mapuniform -- Cells get non spatial value taken from an uniform distribution
max -- Maximum value of multiple expressions
min -- Minimum value of multiple expressions
mod -- Remainder of integer division of values on first expression by values on second expression
ne or != -- Relational-not-equal-to operation
nodirection -- Expression of directional data type
nominal -- Data conversion data type nominal data type
normal -- Boolean TRUE cell gets value taken from a normal distribution
not -- Boolean-NOT operation
or -- Boolean-OR operation
order -- Ordinal numbers to cells in ascending order
ordinal -- Data conversion to the ordinal data type
path -- Path over the local drain direction network downstream to its pit
pit -- Unique value for each pit cell
plancurv -- Planform curvature calculation using a DEM
pred -- Ordinal number of the next lower ordinal class
profcurv -- Profile curvature calculation using a DEM
rounddown -- Rounding down of cellvalues to whole numbers
roundoff -- Rounding off of cellvalues to whole numbers
roundup -- Rounding up of cellvalues to whole numbers
scalar -- Data conversion to the scalar data type
sin -- Sine
slope -- Slope of cells using a digital elevation model
slopelength -- Accumulative-friction-distance of the longest accumulative-friction-path upstream over the local drain direction network cells against waterbasin divides
spread -- Total friction of the shortest accumulated friction path over a map with friction values from source cell to cell under consideration
spreadldd -- Total friction of the shortest accumulated friction downstream path over map with friction values from an source cell to cell under consideration

spreadlddzone -- Shortest friction-distance path over map with friction from a source cell to cell under consideration, only paths in downstream direction from the source cell are considered
spreadmax -- Total friction of the shortest accumulated friction path over a map with friction values from a source cell to cell under consideration
spreadzone -- Shortest friction-distance path over a map with friction from an identified source cell or cells to the cell under consideration
sqr -- Square
sqrt -- Square root
streamorder -- Stream order index of all cells on a local drain direction network
subcatchment -- (Sub-)Catchment(s) (watershed, basin) of each one or more specified cells
succ -- Ordinal number of the next higher ordinal class
tan -- Tangent
time -- Timestep
timeinput... -- Cell values per timestep read from a time series that is linked to a map with unique identifiers
timeinput -- Set of output maps per timestep with an extension that refers to the time at the timestep
timeoutput -- Expression value of an uniquely identified cell or cells written to a time series per timestep
timeslice -- Timeslice
uniform -- Boolean TRUE cell gets value from an uniform distribution
uniqueid -- Unique whole value for each Boolean TRUE cell
upstream -- Sum of the cell values of its first upstream cell(s)
view -- TRUE or FALSE value for visibility from viewpoint(s) defined by a digital elevation model
windowaverage -- Average of cell values within a specified square neighbourhood
windowdiversity -- Number of unique values within a specified square neighbourhood
windowhighpass -- Increases spatial frequency within a specified square neighbourhood
windowmajority -- Most occurring cell value within a specified square neighbourhood
windowmaximum -- Maximum cell value within a specified square neighbourhood
windowminimum -- Minimum value within a specified square neighbourhood
windowtotal -- Sum of values within a specified square neighbourhood
xcoordinate -- X-coordinate of each Boolean TRUE cell
xor -- Boolean-XOR operation
ycoordinate -- Y-coordinate of each Boolean TRUE cell

B Program of the opgewekt.nu exposition opening

Lezingen op 9/11

Vele sprekers zullen 9 november hun kennis en innovatieve ideëen tentoonspreiden. Maar in de lezingen is er ook ruimte voor jóuw inbreng. Opgewekt.nu is namelijk benieuwd naar álle stemmen in het energiedebat.

Er zullen drie rondes met lezingen zijn de 9e.

Sprekers
Louis Hiddes, Directeur Panagro Vastgoedontwikkeling bv. Voorzitter Projectgroep Duurzame Energie

“Zijn wij niet te laat?”

Je kunt energie besparen en proberen minder CO2 uit te stoten, maar is dat genoeg? Wanneer de prijsstijging van olie en gas doorzetten zijn onze woonlasten in 2030 onbetaalbaar geworden! Minder verbruiken is de eerste opgave die ons nu dus te doen staat.

ing. Pieter van Alphen, directeur Techneco Energiesystemen (levert warmtepompen, zonnepanelen en warmtepopmboilers)

“Hoe werkt een warmtepomp?”

‘Met een warmtepomp kun je je huis zowel verwarmen als koelen.’ Maar hoe dan? ‘Een warmtepomp levert je een flinke energiebesparing.’ Maar hoeveel dan? Antwoorden in de lezing!

Jaap Schellekens, WL | Delft Hydraulics (onafhankelijk instituut voor toegepast onderzoek en gespecialiseerd advies op het gebied van alle aan water gerelateerde vraagstukken)

“Amersfoort aan zee”

Af en toe vermelden onheilspellende berichten dat de kans op overstromingen veel groter is dan door de meeste mensen wordt aangenomen. Hoe reëel zijn deze berichten, én waardoor zullen wij natte voeten krijgen? De zee, de rivier of gewoon door een enorme lokale regenbui?
René Visser, directeur Zero-e b.v. (richt zich op alle alternatieve brandstoffen waaronder aardgas, waterstof en elektriciteit)

“Schoner rijden, de mogelijkheden!”

Het toepassen van schone brandstoffen, wat kan het u opleveren? In deze sessie uitleg over voertuigen, brandstoffen, tankstations en emissies, kortom alles over schone brandstoffen.

Wolf Brings

“De vruchten die een actief energiebeleid afwerpen”

Wat is het resultaat als de regering alternatief brandstofgebruik actief stimuleert, en hoe snel werpt zo’n beleid zijn vruchten af? Een vergelijking van de energiesituatie van Nederland met de situatie in vier andere rijke EU-landen.
Steven Gerritsen, architect Noorderlicht (café-restaurant op het NDSM-terrein)

“Heb respect voor de kou man!”

Misschien kun je meer energie bezuinigen door bij kou een warme trui aan te trekken en even wat rustiger aan te doen dan door je te omringen met 650m2 steenwol, 400m2 vloerverwarming en een 120m diep geslagen warmtepomp met wamtewisselaar, buffervat en electrische installatie?
Ferd Crone, lid van de Tweede-Kamerfractie van de PvdA - MISSCHIEN

“Energie en verkiezingsprogramma’s”
prof. ir. Kees Daey Ouwens, (duurzame) energiedeskundige.

“Plantaardige olie (jatropha) als oplossing voor de armoede én voor het wereldenergieprobleem”
ir. Chris Zijdeveld, Stichting Lage Temperatuur Verwarming, Stichting Passief Bouwen.nl en Stichting HR-Ventilatie.

“Comfortabel en energiezuinig wonen”
Jadranka Cace, Vice-voorzitter van de Organisatie voor Duurzame Energie

“Schonere elektra”

C PcrTransformation Component Document

Client:
WL | Delft Hydraulics

Delft- FEWS System

PcrTransformation Component Document

Version 0.1

December, 2006

Contents

1 Introduction

The PcrTransformation Runnable module is a specific module that allows the user to use all the functionalities of PCRaster within FEWS System. The module can be configured so as to provide complete environmental dynamic modelling functionalities of PCRaster. This is possible by using the PCRaster Modelling Environment (PCRME).

The PCRME is a computer language for construction of iterative spatio-temporal environmental models. PCRaster is a complete dynamic modelling system for distributed simulation models. The main applications are found in environmental modelling such as; geography, hydrology, ecology.

PCRaster is originally designed and developed at Utrecht University, department of Physical Geography and is now supported by the company PCRaster Environmental Software that combines its R&D efforts with Utrecht University.

The functionality of the PcrTransformation module has some constraints:

- Dynamic modelling, which is modelling of processes over time is not available.
- The PcrTransformation module will be called by the Workflow server only.

Internet Sources with information:
PCRaster Official web page
http://www.pcraster.nl/

2 Role in FEWS

The PcrTransformation Module will be the FEWS module used for dynamic environmental modelling. This module basically serves as a wrapper to the PCRaster but with its own (FEWS compatible) configuration file. All functionalities extended by PCRME (pcrme.dll, version September 2006, XML based) are available within FEWS using the PcrTransformation Module.

2.1 PCRaster Modelling Environment

PCRaster Modelling Environment (PCRME) is a high level computer language: which uses spatio-temporal operators with intrinsic functionality especially meant for construction of spatio-temporal models. Compared with low level computer languages (e.g. C, Pascal) this has the advantage that models are programmed and structured according to the way of thinking applied in spatial-temporal sciences such as Geography or Geology. It allows researchers in these fields to construct models by themselves in a relatively short period of time, even when they do not have experience in programming. A list of all space-time functions in the PCRaster Environmental Modelling language can be found in the PCRaster Version 2 Manual. Further, models constructed in the PCRaster Environmental Modelling language can easily be changed or extended.

2.2 Functionality

The Environmental Modelling language of PCRaster provides a set of more than 120 spatial and temporal operations that can be used for building (static) Cartographic Models and Dynamic Models. These include:

- Point operations: analytical and arithmetical functions, Boolean operators, conditional statements, operators for relations, comparison, rounding, (random) field generation.
- Window operations: for calculations in moving windows of variable size (highpass filtering, edge filtering, moving averages, etc.)
- Area operations : for calculations in specified areas or classes
- Spread operations : for calculation of distances or cost paths over a map
- Geomorphological operations : functions for hillslope and catchment analysis, definition of hydrological topology
- Hydrological operations : for modelling transport (drainage) of material over a local drain direction map with routing functions
- Time operations: for retrieving and storing temporal data in iterative Dynamic Models

The PcrTransformation module will be capable of allowing all the above mentioned operations available within PCRaster to the users of this module. Further, the module is designed in such a way so that it will be able to handle operations that can be available in PCRaster in future.

3 Feature List

Nr.	Description	Restriction
1	All functionalities extended by PCRaster Modelling Envrionmental should be possible via FEWS	
2	Define Area Maps using a FEWS Grid Location, defining grid in configuration or by providing Grid TimeSeriesSet	
3	Inputs can be Constant, TimeSeries (Grid or Scalar types) or external file.	
4	Output is TimeSeries (Grid) or external file.	

4 Requirements

There are two types of data required by the PcrTransformation Module:

- Time series values within a specified period. (Grid or Point values);
- XML configuration file.

5 Design

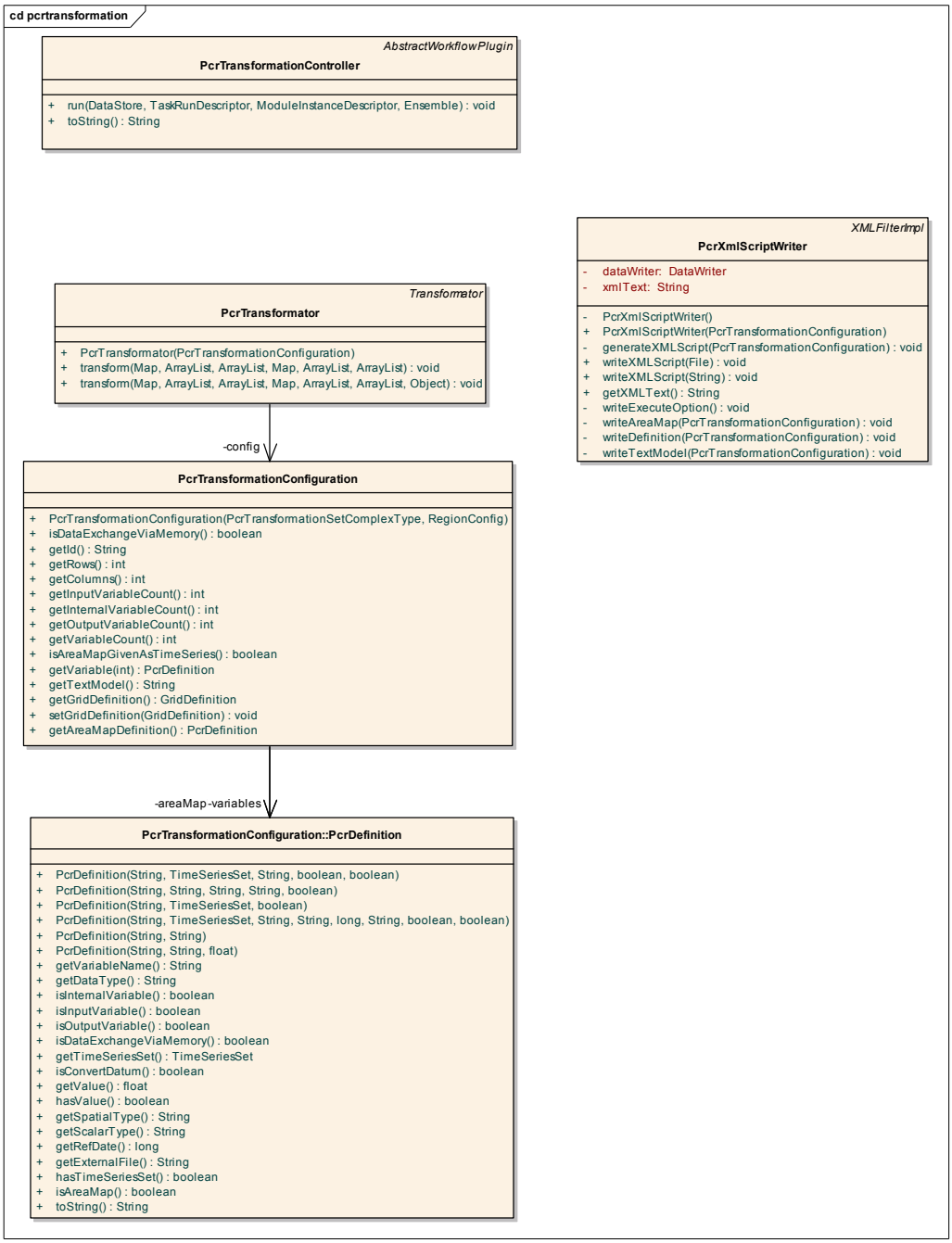
5.1 Introduction

The PcrTransformation Controller component resides in the package nl.wldelft.fews.system.plugin.transformation and consists of a number of important parts:

- *PcrTransformationController*: A main PcrTransformation controller class implementing WorkFlowPlugin runnable.
- *PcrTransformationConfiguration*: An object class which reads the PcrTransformation XML configuration file.
- *PcrTransformator*: A class which prepares the data for PCRaster operations and runs the PCRaster script.
- *PcrXMLScriptWriter*: A class which writes the PCRaster XML script file which is then run using the function calls via PCRaster Modelling Environment Language Dynamic Link Library (pcrme.dll).

5.2 Structure

The figure below shows the UML diagram for PcrTransformation controller.



5.3 Dynamics

The PcrTransformation Controller is a workflow plug-in component. It therefore implements the interface WorkFlowPlugin.

5.4 Event Handling

The PcrTransformation Component will log the following system events:

Event	Description

These system events will be represented by generic system event types that accept the name of the module as (optional) argument.

5.5 Exception handling

The error, within the PcrTransformation Controller, will be logged and control is returned back to WorkFlowPlugin.

6 Implementation

6.1 Issues

Listing of open issues with respect to the design of the utility

Nr	Description / Required Action

6.2 Limitations

Listing of current limitations with respect to the design of the utility

Nr	Description / Required Action
1	Output Variable of is defined as Grid TimeSeries for a single location.
2	Timestepping in pcraster API not yet defined

6.3 Coding Guidelines

7 Configuration

The PcrTransformation Controller is configured using a XML file. This XML configuration file is part of FEWS ModuleConfigFiles (or in other words, this configuration file should be kept under ModuleConfigFiles directory).

The schema required for configurating this module is “pcrTransformationSets.xsd” and available from URL “http:\\fews.wldelft.nl\schemas\version1.0\” Please refer to section 10.2 for schema documentation.

8 Testing

8.1 Unit Testing

8.2 Module Testing

See Q4947 R&D report

9 References

PCRaster Version 2 User Manual
FEWS Configuration Guide

10 Appendix

10.1 Whiteboard sessions

1.1.1 13/07/2006

Present: Jaap Schellekens / Juzer Dhondia / Marc

- Background information and requirements regarding the PCRaster Transformation within FEWS
- Jaap will prepare the initial version of pcrTransformationSets.xsd (XML Schema for PcrTransformation Module) by 24 July.
- Juzer will work on the PcrTransformation Module and before his vacation and submit the initial version for testing to Jaap (Phase 1).
- Basic Requirements are that,
  - Area Map to be defined as
    - location Id (for which the Grid Definition is configured in GridConfig file),
    - number of Rows/Cols
    - grid TimeSeriesSet
  - Input/Output should be defined as
    - Constant
    - External
    - TimeSeriesSet (both Scalar and Grid)
- The Phase 1 would include –
  - Use of PCRaster January 30 2006 version (which includes PCRaster script in XML form + data transfer via Memory)
  - Area Map via Location ID or Grid Definition
  - Input Variable definition via Grid TimeSeries, and possibly support all data types as available within PCRaster. However functionality for defining TimeSeriesSet using LocationSetID is shifted to next phase.
  - Output Variable definition via (no locationSetId) Grid TimeSeries.
- The Phase 2 would include –
  - Area Map from Grid TimeSeries
  - Input Variable definition extended to include constant , external file, scalar time series, time series
- Juzer will do the implementation and unit tests and is responsible to complete user Manual for this transformation.
-

1.2.1 23/09/2006

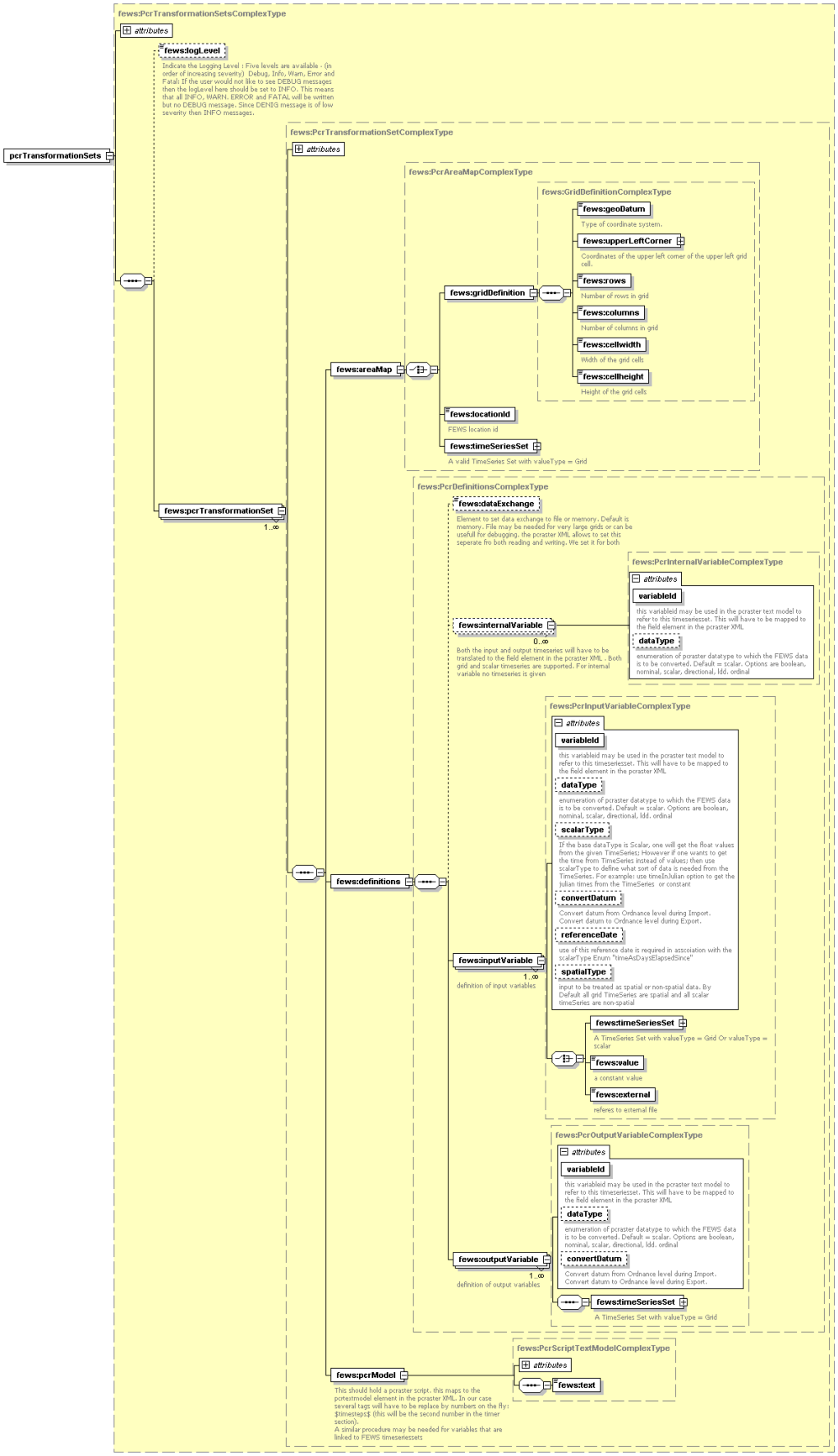
Present: Jaap Schellekens & Juzer Dhondia

Initial Tests performed by Jaap and (unit Tests) by Juzer were successful. The Phase 1 was completed. The work for Phase 2 will be

- Area Map to be defined as
  - grid TimeSeriesSet
- Input/Output should be defined as
  - TimeSeriesSet (both Scalar and Grid)
- Input can be of Type
  - Scalar
  - Non-Scalar
- Possibility to have Time (Date) as input.
  - As Month of Year
  - As Day of Month
  - As Hour of Day
  - Etc.
- Dynamic Time Step
  - **This functionality is present not available within current PCRaster version**

10.2 Schema Documentation

This section describes in detail the pcrTransformationSets.xsd schema file for PcrTransformation Module. Shown below is the schema diagram.

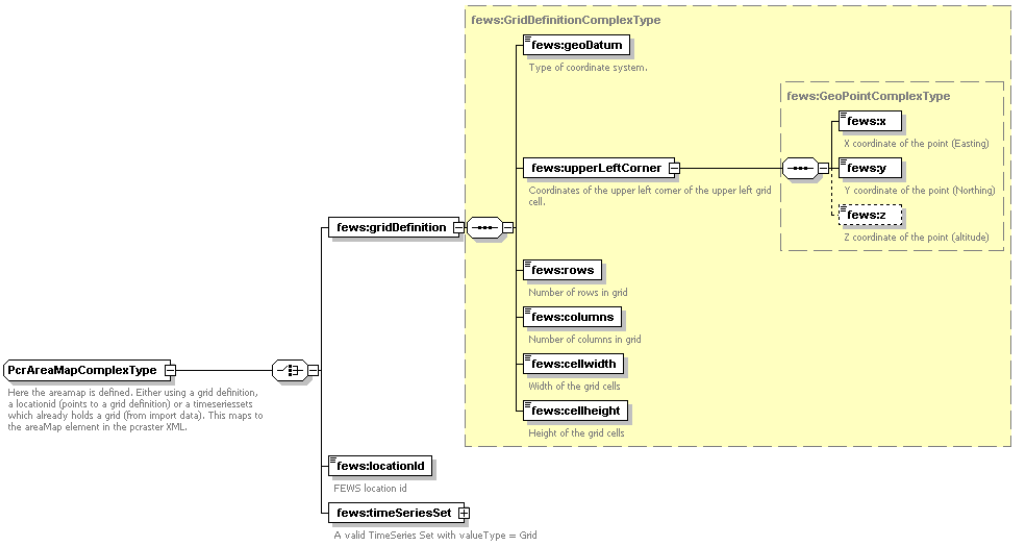

fews:PcrTransformationSetsComplexType
attributes
fews:logLevel
Indicate the Logging Level : Five levels are available - (in order of increasing severity) Debug, Info, Warn, Error and Fatal: If the user would not like to see DEBUG messages then the loglevel here should be set to INFO. This means that all INFO, WARN, ERROR and FATAL will be written but no DEBUG message. Since DENIG message is of low severity then INFO messages.
pcrTransformationSets
fews:PcrTransformationSetComplexType
attributes
fews:PcrAreaMapComplexType
fews:GridDefinitionComplexType
fews:geoDatum
Type of coordinate system.
fews:upperLeftCorner
Coordinates of the upper left corner of the upper left grid cell.
fews:rows
Number of rows in grid
fews:columns
Number of columns in grid
fews:cellwidth
Width of the grid cells
fews:cellheight
Height of the grid cells
fews:gridDefinition
fews:areaMap
fews:locationId
FEWS location id
fews:timeSeriesSet
A valid TimeSeries Set with valueType = Grid
fews:PcrDefinitionsComplexType
fews:dataExchange
Element to set data exchange to file or memory. Default is memory. File may be needed for very large grids or can be usefull for debugging. the pcraster XML allows to set this seperate fro both reading and writing. We set it for both
fews:PcrInternalVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd. ordinal
fews:internalVariable
0..∞
Both the input and output timeseries will have to be translated to the field element in the pcraster XML . Both grid and scalar timeseries are supported. For internal variable no timeseries is given
fews:PcrInputVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd. ordinal
scalarType
If the base dataType is Scalar, one will get the float values from the given TimeSeries; However if one wants to get the time from TimeSeries instead of values; then use scalarType to define what sort of data is needed from the TimeSeries. For example: use timeInJulian option to get the Julian times from the TimeSeries or constant
convertDatum
Convert datum from Ordnance level during Import. Convert datum to Ordnance level during Export.
referenceDate
use of this reference date is required in asscoiation with the scalarType Enum "timeAsDaysElapsedSince"
spatialType
Input to be treated as spatial or non-spatial data. By Default all grid TimeSeries are spatial and all scalar timeSeries are non-spatial
fews:inputVariable
1..∞
definition of input variables
fews:timeSeriesSet
A TimeSeries Set with valueType = Grid Or valueType = scalar
fews:value
a constant value
fews:external
referes to external file
fews:PcrOutputVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd. ordinal
convertDatum
Convert datum from Ordnance level during Import. Convert datum to Ordnance level during Export.
fews:outputVariable
1..∞
definition of output variables
fews:timeSeriesSet
A TimeSeries Set with valueType = Grid
fews:definitions
fews:pcrTransformationSet
1..∞
fews:PcrScriptTextModelComplexType
attributes
fews:text
fews:pcrModel
This should hold a pcraster script. this maps to the pcrestmodel element in the pcraster XML. In our case several tags will have to be replace by numbers on the fly: $timesteps$ (this will be the second number in the timer section).
A similar procedure may be needed for variables that are linked to FEWS timeseriessets

The following sections describe in detail each schema sub-sections. There are three sub-sections:

1. Defining an Area Map;
2. Defining internal, input and output variables;
3. Defining the PCRaster Model.

2.1.1 Defining Area Map

The diagram below gives an overview of schema on how to define an area map.



The area map can be given as:

1. Grid Definition (like – number of rows, columns etc.);
2. (FEWS) Grid location Id;
3. (FEWS) TimeSeriesSet which defines a Grid TimeSeries.

The last option is used when the (FEWS) Grid Location is not configured (in, “RegionConfigFiles/Grids 1.00 default.XML”) but the grid definition is generated programmatically, for example, while importing a grid data from an external grib file.

2.1.1.1 Examples

Here are few examples, which show different possibilities to define an Area Map.

1. The area map is given as Grid Definition. The grid definition contains information such as GeoDatum, coordinates of upper left grid point, number of columns and rows and cell width and height.

```
<areaMap>
        <geoDatum>WGS 1984</geoDatum>
        <upperLeftCorner>
                <x>2</x>
                <y>1</y>
```

```
                    <z>90</z>
                </upperLeftCorner>
                <rows>100</rows>
                <columns>100</columns>
                <cellwidth>0.1</cellwidth>
                <cellheight>0.1</cellheight>
</areaMap>
```

2. The area map is defined as a (FEWS) grid location id which refers to the grid definition at the same location within Grids.xml configuration file

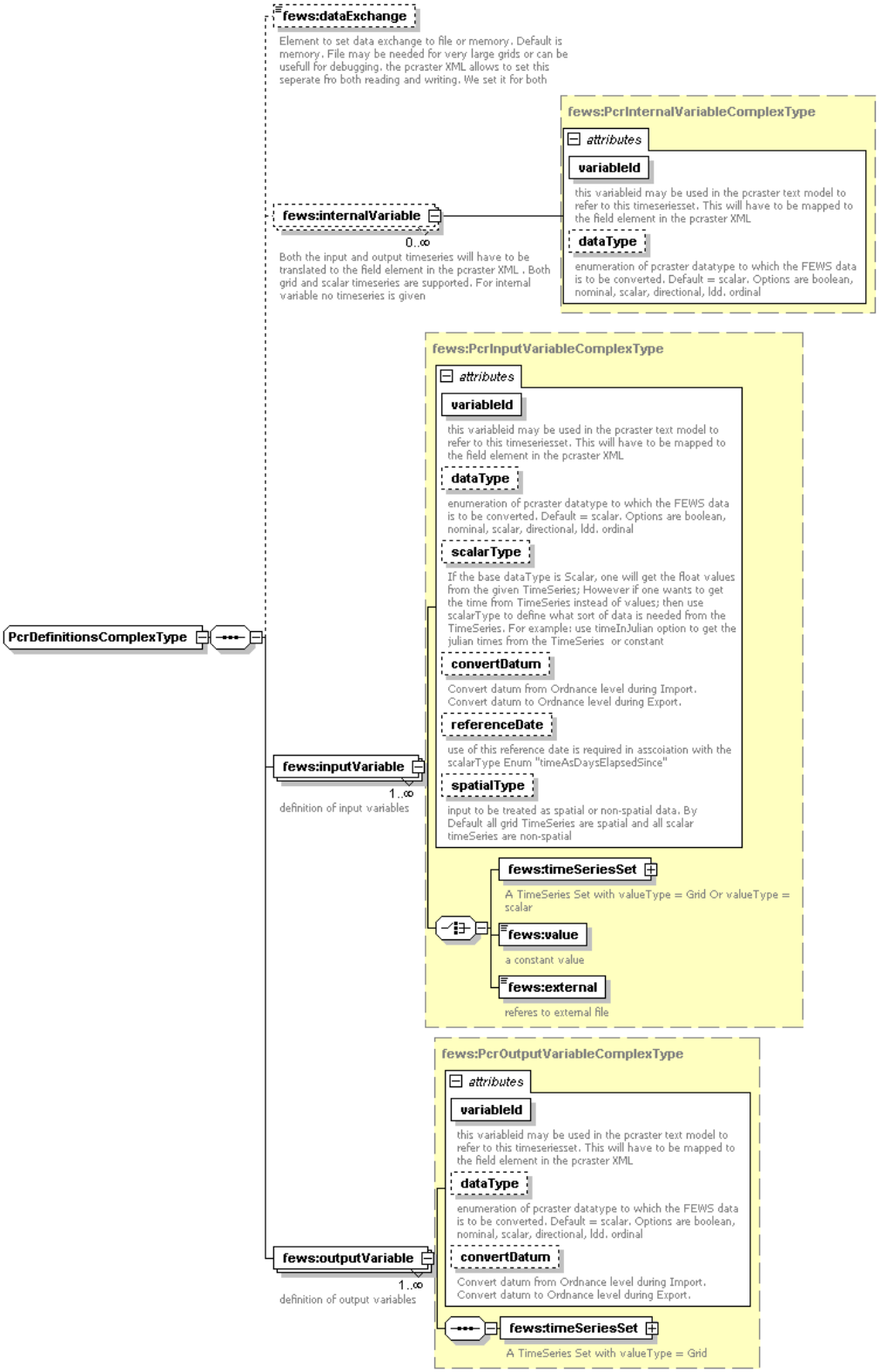
```
<areaMap>
        <locationId>H-2002</locationId>
</areaMap>
```

3. The area map is defined as a (FEWS Grid) TimeSeries Set. For details on how to define TimeSeriesSet please refer FEWS Configuration Guide.

```
<areaMap>
        <moduleInstanceId>ImportGrid</moduleInstanceId>
        <valueType>grid</valueType>
        <parameterId>P.m</parameterId>
        <locationId>MeteoGrid</locationId>
        <timeSeriesType>external historical</timeSeriesType>
        <timeStep unit="hour"/>
        <relativeViewPeriod unit="day" start="0" end="1"/>
        <readWriteMode>read only</readWriteMode>
<areaMap>
```

2.2.1 Defining internal, input and output variables

The diagram below gives an overview of schema on how to define a PCRaster Variables.


PcrDefinitionsComplexType
fews:dataExchange
Element to set data exchange to file or memory. Default is memory. File may be needed for very large grids or can be usefull for debugging. the pcraster XML allows to set this seperate fro both reading and writing. We set it for both
fews:internalVariable
0..∞
Both the input and output timeseries will have to be translated to the field element in the pcraster XML . Both grid and scalar timeseries are supported. For internal variable no timeseries is given
fews:PcrInternalVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd, ordinal
fews:inputVariable
1..∞
definition of input variables
fews:PcrInputVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd, ordinal
scalarType
If the base dataType is Scalar, one will get the float values from the given TimeSeries; However if one wants to get the time from TimeSeries instead of values; then use scalarType to define what sort of data is needed from the TimeSeries. For example: use timeInJulian option to get the julian times from the TimeSeries or constant
convertDatum
Convert datum from Ordnance level during Import. Convert datum to Ordnance level during Export.
referenceDate
use of this reference date is required in asscoiation with the scalarType Enum "timeAsDaysElapsedSince"
spatialType
input to be treated as spatial or non-spatial data. By Default all grid TimeSeries are spatial and all scalar timeSeries are non-spatial
fews:timeSeriesSet
A TimeSeries Set with valueType = Grid Or valueType = scalar
fews:value
a constant value
fews:external
referes to external file
fews:outputVariable
1..∞
definition of output variables
fews:PcrOutputVariableComplexType
attributes
variableId
this variableid may be used in the pcraster text model to refer to this timeseriesset. This will have to be mapped to the field element in the pcraster XML
dataType
enumeration of pcraster datatype to which the FEWS data is to be converted. Default = scalar. Options are boolean, nominal, scalar, directional, ldd, ordinal
convertDatum
Convert datum from Ordnance level during Import. Convert datum to Ordnance level during Export.
fews:timeSeriesSet
A TimeSeries Set with valueType = Grid

Here one can define:

1. Data Exchange Type- *memory* or *file,*

2. Internal PCRaster model variable, with a unique Id and data type. The data type (which are exactly similar to the data types used in PCRaster) can be
   a) boolean
   b) nominal
   c) ordinal
   d) scalar
   e) ldd
   f) directional

3. Input PCRaster model variable. For input variable, one have to provide information, such as
   - Variable id (should be matching exactly as defined in the PCRaster text model),
   - Data type (similar to that used for internal variables),
   - Scalar type data to be passed to PCRaster, if different from the normal data value. At present the following options are available such as,
     – timeInJulian
     – timeAsDayofYear
     – timeAsDayofMonth
     – timeAsHourofDay
     – timeAsDaysElapsedSince
     – timeAsHoursElapsedSince
   - Reference date, provide reference date if scalar type is defined as "timeAsDaysElapsedSince" or "timeAsHoursElapsedSince".
   - Spatial type, has options - spatial and non-spatial. Generally all grid input timeseries are treated as spatial data, while all scalar timeseries (or constant values) are treated as non-spatial. To treat the scalar timeseries (single data value per time) value or constant value as spatial, one can set this option to "spatial". By doing so, the grid (as defined by area map) will be filled with (single) data value from timeseries for a corresponding timestep. Hence for a given timestep, the input to the PCRaster model will be a grid with a constant value in all the grid cells.
   - For spatial type, however there is exception to the above mentioned approach. If the input variable is a scalar timeseries at multiple locations (using LocationSetId in TimeSeriesSet definition) and spatial type is set to spatial, and then the following approach is used,
     1. Create a grid (as defined by area map),
     2. Get the data value for a current time from a timeseries for a first location,
     3. For this location, get the geo-reference coordinates (X, Y),
     4. Get the corresponding grid cell within which the above location lies,
     5. Put the data value as that grid cell value,
     6. Get the data value for the current time from the timeseries for next location.
     7. Repeat the steps 3-6 till all the data for the current time at all locations is read and the value is put in the appropriate grid cell.
   - TimeSeriesSet, if the input data is a timeseries.
   - Value, if the input data is a constant value, and

- External, if the data has to be read from the external PCRaster formatted file.

4. Output PCRaster model variable. For output variable, one have to provide information, such as
   - Variable id (should be matching exactly as defined in the PCRaster text model.
   - Data type (similar to that for internal variables), and
   - TimeSeriesSet, (at present) all output data should be grid timeseries.

2.2.1.1 Examples

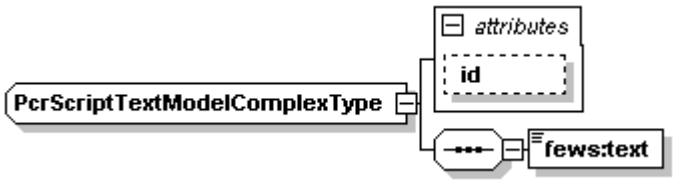
Here are few examples, showing different possibilities to define an interval, input and output variables. Refer to the comments for details as given before each entry.

```
<definitions>
        <!-- dataExchange options = Memory -->
        <dataExchange>memory</dataExchange>

        <!-- internalVariable name used within the PCRaster Test Model -->
        <internalVariable variableId="blnmap" dataType="boolean"/>

        <!-- internalVariable name used within the PCRaster Test Model
                , now with the dataType as scalar -->
        <internalVariable variableId="toSpatial" dataType="scalar"/>

        <!-- InputVariable which refers to the external data file -->
        <inputVariable variableId="externalVar" dataType="scalar">
                <external>d://x.txt</external>
        </inputVariable>

        <!—Input Variable which refers to TimeSeriesGrid Array i.e. Grid as input -->
        <inputVariable variableId="input" dataType="scalar" convertDatum="false">
                <timeSeriesSet>
                        <moduleInstanceId>ModuleInstance</moduleInstanceId>
                        <valueType>grid</valueType>
                        <parameterId>H.tidal</parameterId>
                        <locationId>H-2002</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>
        </inputVariable>

        <!-- InputVariable which refers to the TimeSeries Float Array i.e, scalar value per
                time, non spatial in nature . In other words, a value per time distributed
                constantly over the whole grid for calculation purpose -->
        <inputVariable variableId="input" dataType="scalar" convertDatum="false">
                <timeSeriesSet>
                        <moduleInstanceId> ModuleInstance</moduleInstanceId>
                        <valueType>scalar</valueType>
                        <parameterId>H.tidal</parameterId>
                        <locationId>H-2002</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>
        </inputVariable>

        <!-- InputVariable which refers to the Constant Value i.e, a constant scalar value
                irrespective of time and non spatial in nature. In other words, a constant
```

```
        value distributed constantly over the whole grid for calculation purpose -->
<inputVariable variableId="constant" dataType="scalar">
        <value>10</value>
</inputVariable>

<!-- InputVariable which refers to the TimeSeries Float Array for multiple location
        (given by locationSetID) i.e,  scalar value per time, spatial in nature (as
        given by spatialType),  and  defined only at grid cells where contains the
        location. In other words, the  grid cell which contains the georeference
        position of the location -->
<inputVariable variableId="input" dataType="scalar"
        spatialType="spatial" convertDatum="false">
        <timeSeriesSet>
                <moduleInstanceId>ModuleInstance</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>H.tidal</parameterId>
                <locationId>TestLocLiesWithinGrid_H-2002</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="1"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</inputVariable>

<!-- InputVariable which refers to the TimeSeries Float Array  (NonSpatial) ,
        however  the time is passed to PCRaster  (scalarType =
        timeAsDaysElapsedSince).    For the scalar Type defined as time "Elapsed
        Since" the reference Date  has to be defined  -->
<inputVariable variableId="input" dataType="scalar"
                scalarType="timeAsDaysElapsedSince"
                referenceDate="1984-10-30" convertDatum="false">
        <timeSeriesSet>
                <moduleInstanceId>ModuleInstance</moduleInstanceId>
                <valueType>scalar</valueType>
                <parameterId>H.tidal</parameterId>
                <locationId>H-2002</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="1"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</inputVariable>

<!—Output Variable Using which refers to a TimeSeries  Grid Array -->
<outputVariable variableId="transfmap" dataType="scalar" convertDatum="false">
        <timeSeriesSet>
                <moduleInstanceId>OutputModuleInstance</moduleInstanceId>
                <valueType>grid</valueType>
                <parameterId>H.updated</parameterId>
                <locationId>H-2003</locationId>
                <timeSeriesType>external historical</timeSeriesType>
                <timeStep unit="day"/>
                <relativeViewPeriod unit="day" start="0" end="1"/>
                <readWriteMode>add originals</readWriteMode>
        </timeSeriesSet>
</outputVariable>
```

2.3.1 Defining the PCRaster Model

The diagram below gives an overview of schema on how to define a PCRaster Model.



In this section, one can provide the PCRaster model as simple ASCII text. The text model given here is in fact the valid PCRaster model and can be run directly using PCRaster model, except that it does not contain any area map or variable definition part. All the variables used within this model should appear in the definition section as described in section 2.2.1 above.

2.3.1.1 Example

Here is an example, showing how to configure a PCRaster Model.

```
        <pcrModel id="String">
                <text>
<!—PCRaster accepts #  as Comment—>
# there is no dynamic section!
#initial

#passed
#result should be the grid within constant value of 1.8 and 0.8
# generate unique Id's
Unq = uniqueid(boolean(input));
# Now generate polygons
transfmap = spreadzone(ordinal(cover(Unq,0)),0,1);
= areaaverage(input,GaugeArea);

                </text>
        </pcrModel>
```

Please remember, the PCRaster model, which is defined here, is full PCRaster model script written in PCRaster Modelling Environment language. Take care that all the variables ids defined in Variable definition section matches the variables used here in the model. In other words, the model defined here should be a PCRaster compatible script.

10.3 Sample configuration to perform a typical PCRaster Transformation

A simple working sample configuration for PcrTransformation is shown as below.

```
<?xml version="1.0" encoding="UTF-8"?>
<pcrTransformationSets xmlns="http://www.wldelft.nl/fews"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.wldelft.nl/fews:\nffs\Implementation\SOFTWARESVN\JAVA\xml-
schemas\pcrTransformationSets.xsd" version="1.1">
    <pcrTransformationSet id="PCr">
            <!--  Defining AreaMap -->
            <!--  Here the area map is defined as a location id which refers to the grid definition at same
location within grids.xml file -->
        <areaMap>
            <locationId>H-2002</locationId>
```

```
</areaMap>
 <!--  Another options are  providing the area map as  number of columns and rows -->
 <!--  and by providing the TimeSeries Set of  type 'grid' which contains the grid definition. -->
<definitions>
        <!--  dataExchange  options =  Memory -->
        <dataExchange>memory</dataExchange>
        <!--  internalVariable  name used within the PCRaster Test Model -->
        <internalVariable variableId="blnmap" dataType="boolean"/>
        <!--  internalVariable name used within the PCRaster Test Model
                , now with the dataType as scalar -->
        <internalVariable variableId="toSpatial" dataType="scalar"/>
        <!--  InputVariable which refers to the external data file -->
        <inputVariable variableId="externalVar" dataType="scalar">
                <external>d://x.txt</external>
        </inputVariable>

        <!-- Using Grid as Input -->
        <inputVariable variableId="input" dataType="scalar" convertDatum="false">
                <timeSeriesSet>
                        <moduleInstanceId>ModuleInstance</moduleInstanceId>
                        <valueType>grid</valueType>
                        <parameterId>H.tidal</parameterId>
                        <locationId>H-2002</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>
        </inputVariable>

        <!—OuputVariable (Only Grid + Spatial allowed)  -->
        <outputVariable variableId="transfmap" dataType="scalar" convertDatum="false">
                <timeSeriesSet>
                        <moduleInstanceId>OutputModuleInstance</moduleInstanceId>
                        <valueType>grid</valueType>
                        <parameterId>H.updated</parameterId>
                        <locationId>H-2003</locationId>
                        <timeSeriesType>external historical</timeSeriesType>
                        <timeStep unit="day"/>
                        <relativeViewPeriod unit="day" start="0" end="1"/>
                        <readWriteMode>add originals</readWriteMode>
                </timeSeriesSet>
        </outputVariable>

        <pcrModel id="String">
                <text>
# there is no dynamic section!
#initial
#original

<!—When Using Grid as input -->
#passed
#result should be 1.3 for First TimeStep and  NAN  for second TimeStep.
#  For the all grid cells values for the Input Grid for the first time step is 1.8 and for second timestep is 0.8.
# rest the answer is then  self-explanatory.
transfmap = scalar(if(input  gt 1 then input-0.5));

                </text>
        </pcrModel>
</pcrTransformationSet>
</pcrTransformationSets>
```