Contents

[Usage 2](#_Toc194400307)

[Execution 2](#_Toc194400308)

[DLI 3](#_Toc194400309)

[Pumping 3](#_Toc194400310)

[Overview 3](#_Toc194400311)

[Theory and Equation 4](#_Toc194400312)

[Pumping Code 5](#_Toc194400313)

[Shading 7](#_Toc194400314)

[Overview 7](#_Toc194400315)

[Tree Height Shading Algorithm 7](#_Toc194400316)

[Usage 7](#_Toc194400317)

[Algorithm 7](#_Toc194400318)

[LAI Shading Algorithm 8](#_Toc194400319)

[Usage 8](#_Toc194400320)

[Algorithm 9](#_Toc194400321)

[Code 9](#_Toc194400322)

[Qualité Time Steps 10](#_Toc194400323)

[Overview 10](#_Toc194400324)

[Example Usage: 10](#_Toc194400325)

# Usage

The function is called in the following way:

[outputs] = CequeauQuantiteMex(inputs)

[etatsCE, etatsCP, etatsFonte, etatsEvapo, etatsBarrage, pasDeTemps, avantAssimilationsCE, avantAssimilationsFonte, avanAssimilationEvapo, etatsQualCP, avAssimQual] = **CequeauQuantiteMex**(execution, parametres, bassinVersant, meteo, …

etatsPrecedents, assimilations, stations);

Inputs :

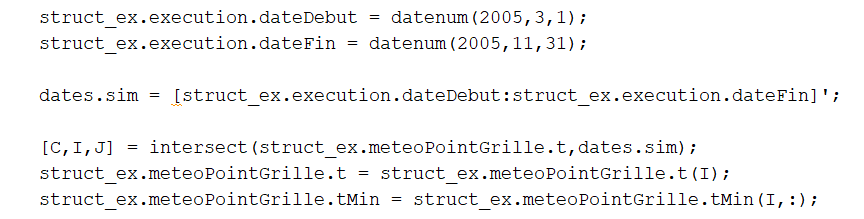
* execution
* parametres
* bassinVersant
* meteo
* etatsPrecedents
* assimilations
* stations

# Execution

The execution struct contains the start and end dates for the simulation. These dates can be modified to simulate a specific subset of the overall data range. However, it is important that the input data matches the specified range.

This is important for:

* All **meteo** data
* **barrage** data (bassinVersant.barrage): **debit**
* **Puits** data (bassinVersant.puits): **debitPompage** and **niveauxPuits**

The intersect function in Matlab/Octave can find the matching date range, which can then be used to extract the relevant dates, as shown below:

# DLI

DLI parameters need to be added within the structure in the latest version of Cequeau. The mat file dli\_params.mat is included in the cequeau/tests/mat\_files folder.

% parametres pour DLI  
load('dli\_params.mat');  
structSMA.parametres.dli = dli\_params;

# Pumping

## Overview

The pumping module takes into account the water being pumped out of the ground for each whole square (CE) at each timestep. Pumping data is provided through data about wells located on the river basin.

To turn on the pumping module, the option must be set to 1. Setting it to 0 (default) will turn it off.

StructFinal.parametres.option.modulePompage = 1;

Data about water wells is placed within the bassinVersant struct. Each well needs to have the index of the CE, the distance from the well to the river, the initial water level h0, all the water levels and the pumping rate in m3 per day. There is also a variable to activate/deactivate the well.

% Initialize the structure to hold all wells  
structPuits = struct();  
   
structPuits(1).idCE = 41; % index of whole square  
structPuits(1).active = 1; % activate the well  
structPuits(1).distanceRiviere = 0; % distance between well and river structPuits(1).h0 = 20; % initial water level structPuits(1).niveauxPuits = ones(numTimeSteps, 1); % water level at each timestep  
structPuits(1).debitPompage = Debit\_Pomp.Well\_1; % pumping rate m3/day at each timestep  
   
structPuits(2).idCE = 1;   
structPuits(2).active = 0; % deactivate the well  
structPuits(2).distanceRiviere = 0; % distance between well and river structPuits(2).h0 = 20; % initial water level structPuits(2).niveauxPuits = ones(numTimeSteps, 1); % water level at each timestep  
structPuits(2).debitPompage = Debit\_Pomp.Well\_2; % pumping rate m3/day at each timestep  
   
StructFinal.bassinVersant.puits = structPuits;

Other parameters are set within the parameters struct.

The “delai” parameter delays the effect of the pumping by the given number of timesteps.

The conductiviteHydraulique\_s is used to specify the hydraulic conductivity. A single value can be given to assign the same values to all whole squares, or a 1 x number of CE array can be provided to assign a value to each CE.

The coeffPompage is used to adjust the extracted water from the reservoir (to be calibrated)

% creating pumping parameter struct  
 StructFinal.parametres.pompage = struct();  
 StructFinal.parametres.pompage.delai = 1;  
 StructFinal.parametres.pompage.coeffPompage= 0.01;   
   
% 1) a single value  
 StructFinal.parametres.pompage.conductiviteHydraulique\_s = 1;  
   
% 2) a value for each CE, the index indicates the idCE [1 x numCE]  
 StructFinal.parametres.pompage.conductiviteHydraulique\_s = ones(1, numel(StructFinal.bassinVersant.carreauxEntiers));

### Theory and Equation

To take into account the wells distance from the river, the radius of influence is calculated for each well during pumping. The following variables and units are used:

*pumped discharge: Qi [m3/day]*

*hydraulic conductivity: K [m/day]*

*Initial head in the pumping well: h0 [m]*

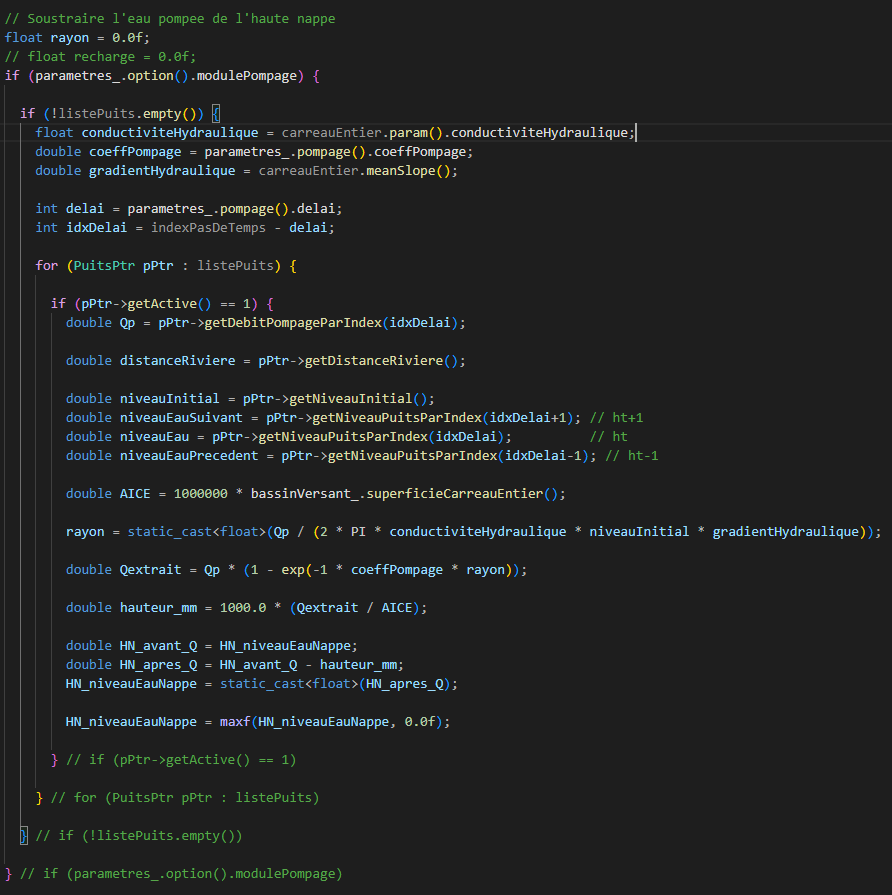
*hydraulic gradient: β*

*radius of influence: r0 [m]*

The hydraulic gradient is approximated as the mean surface slope of the whole square.

To adjust the volume of water removed from the reservoir, *coeffPompage* () is used.

### Pumping Code



# Shading

## Overview

There are two algorithms used to take into account the effects of shading from trees. To activate and choose the algorithm, the struct.parametres.option.moduleOmbrage option is used:

* moduleOmbrage = 0: No shading algorithm is used.
* moduleOmbrage = 1: Shading algorithm based on tree height is used.
* moduleOmbrage = 2: Shading algorithm based on leaf area index (LAI) is used.

## Tree Height Shading Algorithm

### Usage

* Set moduleOmbrage = 1
* Add average tree height for each CP: struct.bassinVersant.carreauxPartiels.hautMoyenneArbre

The following example shows the average tree height for each CP being set to 12 meters.

structSMA.parametres.option.moduleOmbrage = 1;  
 [structSMA.bassinVersant.carreauxPartiels(1:end).hautMoyenneArbre] = deal(12);  
  
 [y1.etatsCE, y1.etatsCP, y1.etatsFonte, y1.etatsEvapo, y1.etatsBarrage, y1.pasDeTemps,...  
 y1.avantAssimilationsCE, y1.avantAssimilationsFonte, y1.avantAssimilationsEvapo,y1.etatsQualCP, y1.avAssimQual] = ...  
cequeauQuantiteMex(structSMA.execution, structSMA.parametres, structSMA.bassinVersant, ...  
 structSMA.meteoPointGrille, [], [], [], []);

### Algorithm

The following is calculated for each partial square (CP)

1. Extract data: latitude, longitude, river width, orientation of the river, tree height, and date.
2. Calculate solar position using SPA algorithm ([NREL’s Solar Position Algorithm for Solar Radiations Applications](https://midcdmz.nrel.gov/spa/#agree))
3. Difference between the sun’s orientation and the river’s orientation
4. Calculate average shadow length based on average tree height
5. Calculate average shadow length perpendicular to the river
6. Shadow ratio covering the river

## LAI Shading Algorithm

### Usage

* Set moduleOmbrage = 2
* Obtain LAI data
  + Data should be in the same format as weather data: [timesteps x CE]
* Normalize LAI data using the provided normalize\_lai MATLAB function
* Add normalized data into the struct: struct.meteoPointGrille.lai\_norm

% normalize LAI values based on given percentile  
percentile = 75;  
structSMA.meteoPointGrille.lai\_norm = normalize\_lai(lai\_data, percentile);  
  
structSMA.parametres.option.moduleOmbrage = 2;  
  
[y2.etatsCE, y2.etatsCP, y2.etatsFonte, y2.etatsEvapo, y2.etatsBarrage, y2.pasDeTemps,...  
 y2.avantAssimilationsCE, y2.avantAssimilationsFonte, y2.avantAssimilationsEvapo,y2.etatsQualCP, y2.avAssimQual] = ...  
cequeauQuantiteMex(structSMA.execution, structSMA.parametres, structSMA.bassinVersant, ...  
 structSMA.meteoPointGrille, [], [], [], []);

### Algorithm

1. LAI data is normalized in a range from 0 to 1, based on a given percentile value.
2. The normalized LAI value is used to scale crayso (struct.parametres.qualite.cequeau.temperat.crayso)
3. The scaled crayso value is used to calculate solar radiation.

### Code

The normalized LAI value is reduced to a maximum of 0.8 to prevent the scaled\_crayso to be reduced to less than 0. The scaled\_crayso is then used to calculate the solar radiation.

# Qualité Time Steps

## Overview

To use sub-daily time steps with the Cequeau Qualité module, the ipassim variable in the input struct’s option must be modified. The value can be set to 1, 2, 3, 4, 6, 8, 12, and 24 (daily). This will change the time steps for both the Quantité and Qualité module.

InputStruct.parametres.option.ipassim = 12;

*Note: data assimilation is only available for daily time steps.*

The calculQualite option must also be enabled to use the Qualité module.

InputStruct.parametres.option.calculQualite = 1;

It is also important to match simulated date range (Struct.execution.dateDebut and Struct.execution.dateFin) to align with the number of data points in the input structures, such as for meteorological data and dam (barrage) data.

## Example Usage:

The file cequeau/run/runCequeau\_Qualite.m demonstrates examples.

For now, testing file is linearly interpolating the input data (processInput.m) to match the number of data points to test the sub-daily water temperature model.

% every 12 hours

InputStruct12 = processInput(InputStruct, indices, dateDebut, dateFin, 12);

[y12.etatsCE, y12.etatsCP, y12.etatsFonte, y12.etatsEvapo, y12.etatsBarrage, y12.pasDeTemps,...

y12.avantAssimilationsCE, y12.avantAssimilationsFonte, y12.avantAssimilationsEvapo, y12.etatsQualCP, y12.avAssimQual] = ...

cequeauQuantiteMex\_test(InputStruct12.execution, InputStruct12.parametres, InputStruct12.bassinVersant, ...

InputStruct12.meteoPointGrille, [], [], [], []);

The same process was used to also simulate using a time step of 6. The following graphs demonstrate the results compared to actual water temperature from stations.

