**Internal Report**

# CORILIS Methodology

## Smoothing of CORINE Land Cover Data

**Prepared by:**
Ferran Páramo

**January 2008**

**Project manager:**
Jean-Louis Weber

European Topic Centre
Land Use and Spatial Information

European Environment Agency

Universitat Autònoma de Barcelona
Edifici C – Torre C5 4ª planta
08193 Bellaterrra (Barcelona)
Spain, EU

Contact: etclusi@uab.cat

# TABLE OF CONTENTS


**1 Overview ..... 4**

**2 Procedure Steps ..... 5**

2.1. Process Overview ..... 5
2.2. Preparing Input data for MatLab ..... 6
2.3. Processing data in MatLab ..... 7
2.3.1. The Smoothing Algorithm ..... 8
2.4. Importing MatLab outputs ..... 9
2.4.1. Preparing layers for the EEA Dataservice ..... 9

**3 Further development ..... 10**

**4 References ..... 11**

**5 Scripts ..... 12**

5.1. PreMatlab.aml ..... 12
5.2. Run.m ..... 13
5.3. CorilisFloat.m ..... 13
5.4. createWindow.m ..... 14
5.5. PostMatlab.aml ..... 15

# 1 OVERVIEW

Spatial smoothing has been proved to be a suitable technique for Land Cover data generalization and analysis.

CORILIS, from CORIne and LISsage (smoothing in French), is a methodology developed jointly by the French Environment Institute (IFEN), the Hypercarte Research Group and the French National Institute for Statistics and Economic Studies (INSEE) that provides technical specifications for the smoothing of CORINE Land Cover Data.

The purpose of CORILIS is to calculate "intensities" or "potentials" of a given theme in each point of a territory. A Gaussian type statistical function (called BiWeight) is used to weight this information according to the distance from the considered point in kilometers.

CORILIS results into probability surfaces (varying from 0 to 100) for the presence of a certain CLC class within a smoothing radius. Individual CORILIS layers from a given level can be aggregated to upper levels by simple addition.

First CORILIS tests used hexagon tessellation as the starting generalization for the smoothing process. This first generalization is needed to reduce the weight of raw data. Direct implementation of smoothing algorithms on pixels of 100 m is not feasible due to the heaviness of the computation.

Later tests carried out at ETCTE changed the hexagon tessellation by regular square grids. Last CORILIS implementation has used the recently defined European Reference Grid of 1 x 1 km as generalization framework (see Table 1).

**Table 1.— History of CORILIS methodology implementations using various software's and data sources.** First implementation was using hexagon tessellation that changed to square reference grids in the following implementations. The attempt of using raw CLC data (100 m pixels) as input for smoothing resulted into a very time consuming process.

| Developer | Land Cover | Tessellation | Software |
|---|---|---|---|
| IFEN: Lacaze (2000) | Raster 250 m | Hexagons (6km base) | SAS |
| ETCTE: Salvador, R. (2001) | Raster 100 m | 100 m pixels | C++ |
| ETCTE: Páramo, F. (2002) | Raster 100 m | 3 Km square cells | Visual Basic 6 |
| ETCTE: Zapata, J.L. (2005) | Raster 100 m | 1 Km square cells | MatLab 7 |

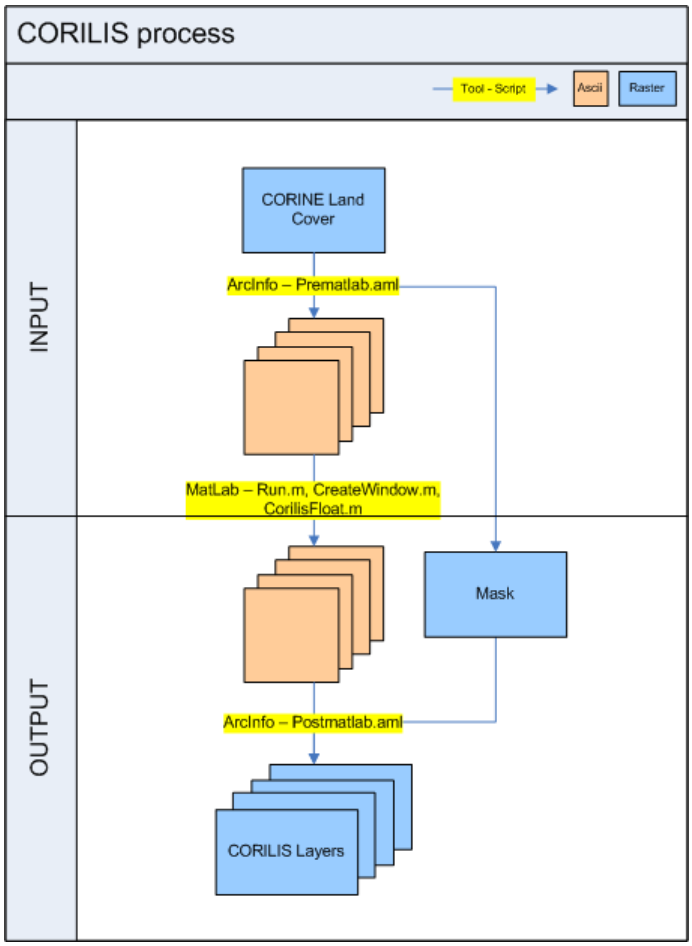
# 2 PROCEDURE STEPS

## 2.1. PROCESS OVERVIEW

The process is carried out in a workspace with two folders, Input and Output. The process can be divided into three main steps depending on the script or tool used:

- **Preparing input data for MatLab:** In this step we extract data from CORINE Land Cover to a set of ASCII files using the script *Prematlab.aml*. This script is written in Arc Macro Language (AML) and is run under ArcInfo Workstation in the INPUT folder. The resulting ASCII files are stored in the same folder. Another output of this step is a mask in grid format that is stored in the folder OUTPUT to be used in a later step.
- **Processing data in MatLab:** The ASCII files produced in the previous step are used as input for MatLab. In this step three scripts are used: Run.m, CreateWindow.m and CorilisFloat.m. The output is a set of ASCII files that will be stored in the folder OUTPUT.
- **Processing MatLab outputs:** The ASCII files produced by MatLab are processed by running the script Postmatlab.aml with ArcInfo Workstation in the OUTPUT folder. This script uses the raster mask created in the first step to clip the outputs.

The figure below shows the different Inputs, Outputs, Tools and Scripts used in the process. Each step is described in detail in the following sections.



## 2.2. PREPARING INPUT DATA FOR MATLAB

First of all CORINE Land Cover (raster 100 m) is converted from TIFF to Arc Info GRID format in order to speed up some of the processing steps. The resulting grid is stored in the INPUT folder.

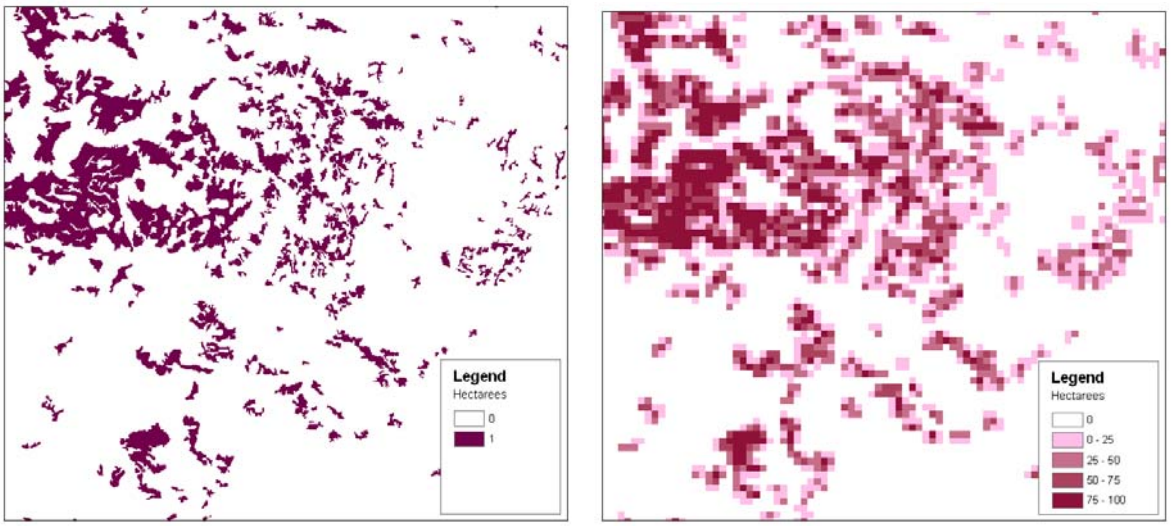
The rest of the process is automated with the script PREMATLAB.aml (see point 5.1). This script performs the following tasks:

- Splitting CLC into 44 individual classes and aggregating each class to a new resolution of 1 km by summing up the number of hectares (see Figure 2.2.1).
- A new class called LAND is created by excluding Sea and Oceans and No Data. This new class LAND is also aggregated to 1 km.
- The LAND grid is reclassified into Boolean values (1: Land, 0: No Land) to create a MASK that will be used in a later step.
- The resulting grids (44 CLC classes + Land) are exported to ASCII format. The ASCII files contain a header with information about the extent of the geographic layer as well as the resolution for the image pixels:

```
<ncols xxx>
<nrows xxx>
<xllcenter xxx | xllcorner xxx>
<yllcenter xxx | yllcorner xxx>
<cellsize xxx>
{nodata_value xxx}
row 1
row 2
.
.
.
row n
```

- These 45 ASCII files are kept in the INPUT folder and will be used as input for the MatLab script.

**Figure 2.2.1. — Aggregating CORINE data.** To perform the analysis at European scale CLC data is generalized to 1x1 km resolution. The aggregation algorithm sums up the number of hectares of a given land cover class in each 1 km cell. The use of data at a resolution of 100 m could only be faced at regional scales (see Chapter 3).



## 2.3. PROCESSING DATA IN MATLAB

The smoothing algorithm (see point 2.3.1.) was implemented in MatLab software which allows high processing rates when working with large datasets. However, the chosen format for input data can represent a limitation for the process (see Chapter 3).

Three scripts are provided in Chapter 5: **run.m**, **corilisFloat.m** and **createWindow.m**. The script **run.m** calls the script **CorilisFloat.m** which performs the main process. The script **CorilisFloat.m** calls the script **CreateWindow.m** to set up the moving window that will be used to smooth the data (see figure 2.3.1),

The use of the MatLab scripts is described in 3 steps:

1. Opening MatLab and setting up the path where the scripts are stored, for example, C:\WORKSPACE\SCRIPTS.
2. Editing the script **run.m** and modifying the following parameters:
    a. **DataPrefix:** The prefix given to the ASCII files.
    b. **InputDir:** The path to the folder with the ASCII files
    c. **OutputDir:** The path where the results will be stored
    d. **AreaFile:** The path to the Totals file.
    e. **R:** The radius in map units (in this case Kilometers)
3. Execute the script **run.m** by typing run().

MatLab will create an ASCII output for each of the 44 classes. These ASCII files will be placed in the Output directory.

**Figure 2.3.1.— The script createWindow.m** defines the moving window that will be used to smooth the data. We can test this script by sending a radius as parameter.

```
>> createWindow(2)

ans =

    0.2500    0.5625    0.2500
    0.5625    1.0000    0.5625
    0.2500    0.5625    0.2500
```

**Figure 2.3.2— Imshow().** We can use this standard function of MatLab to visualize the window created by the createWindow.m script.

```
>> imshow(createWindow(50))
```

## 2.3.1. The Smoothing Algorithm

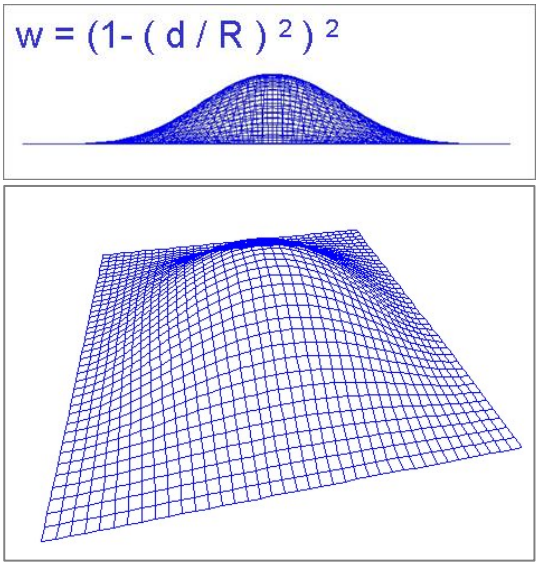
The spatial smoothing consists on determining for each point of the land the potential information present in its neighborhood. A Gaussian type statistical function (called BiWeight) is used to weight this information according to the distance **d** from the considered point in kilometers (see Figure 2.3.1.1). For each CLC class **i** we obtain a probability surface that shows the weighted percentage of that class within a specific radius **R**.

In the aggregation process (Figure 2.2.1) we obtain the raw surface ($\boldsymbol{\sigma_{ij}}$) for each class **i** in each cell **j**. We obtain as well the total land surface in each cell $\boldsymbol{\varphi_j}$

For each class **i** in the cell **j** we obtain its smoothed surface $S_{ij} = \Sigma_\lambda (\sigma_{i\lambda} * w_\lambda)$ where $w_\lambda = (1-(d_\lambda/R)^2)^2$ and $d_\lambda < R$. We also obtain the total smoothed surface $\Phi_j = \Sigma_\lambda (\varphi_j * w_\lambda)$.

With these two indicators we can calculate the relative weight (density indicator) of each class in each cell $\prod_{ij} = 100 * S_{ij} / \Phi_j$. This weight is expressed as percentage in order to allow an easiest reading and interpretation of the results.

**Figure 2.3.1.1.— The Gaussian function** used to weight land cover values in the smoothing process. Where: w is for weight, d for distance in Km and R for radius in kilometers.



## 2.4. IMPORTING MATLAB OUTPUTS

MatLab outputs are stored as ASCII files and can be imported directly to ArcGIS. The preparation of the MatLab outputs is automated in an AML script named PostMatlab.aml (see point 5.5) which is run through ArcGis Workstation in the OUTPUT directory.

This script performs the following tasks:

- Importing the 44 ASCII files to GRID format with the FLOAT option.
- Summing up the 44 grids in a new grid called SUMA.
- Adjusting each of the 44 grids to a new range 0-100.
- Converting each grid to integer in order to reduce the size of the files. This conversion introduces an error due to the nature of data (See table 2.4.1)
- Clipping each grid with the mask produced in the first step in order to eliminate the edge effect produced by the smoothing process.

After running the script Postmatlab.aml we get a set of 44 grids with integer values. Each grid corresponds to a CORINE Land Cover Level 3 class smoothed with a specific radius. These grids can be summed up to produce Levels 2 and Level 1 classes as well as other customized aggregations like "wetlands" or "high nature value" areas.

### 2.4.1. Preparing layers for the EEA Dataservice

Before uploading CORILIS layers to the EEA Dataservice, grids are exported to TIFF format with LZW compression. The projection is defined and metadata created in order to fit the EEA GIS guidelines.

The naming of the files is changed using a simple naming convention that gives more information about its contents. The naming of Corilis layers follows the pattern CORILIS??_R??L?_C?? where R is for Radius, L is for Level and C is for Class.

Level 3 classes are equivalent to the Level 3 classes of Corine. Level 1 classes are a special grouping defined for Environmental Accounting purposes.

For example: CORILIS90_R05L3_C1 contains the smoothing results of CLC90 level 3, class 1 "Continous urban fabric" in a 5 km radius. For each pixel of the grid we get a value (ranging from 0 to 100) indicating the % of that class within the radius of 5 km.

Each set of CORILIS layers (same level and radius) is then zipped and uploaded to the EEA Dataservice: http://dataservice.eea.europa.eu/dataservice/ [Accessed: 2008-02-11]

**Table 2.4.1 — Converting values from float to integer** reduces the size of the files but produces a loose of information. After converting the 44 layers to integer the TOTAL does not match the expected value of 100. In the table below we show and example using 5 layers. The sum of the float values is 100 but after transforming the values to integer we obtain a lower or higher value depending on the method used. Method 1 uses simple truncation while Method 2 assigns the closest integer.

| Layer | Float | Integer Method 1 | Integer Method 2 |
|---|---|---|---|
| Layer 1 | 25.6 | 25 | 27 |
| Layer 2 | 53.3 | 53 | 53 |
| Layer 3 | 12.8 | 12 | 13 |
| Layer 4 | 8.0 | 8 | 8 |
| Layer 5 | 0.3 | 0 | 0 |
| **Total** | **100** | **98** | **101** |

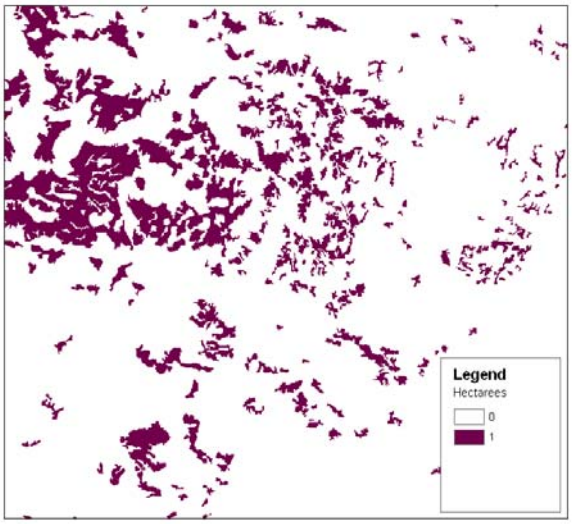
# 3 FURTHER DEVELOPMENT

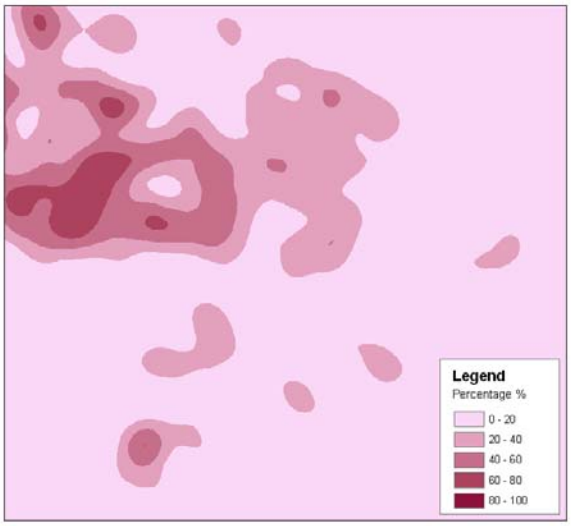
The bottle neck of the smoothing process in MatLab is to read/write data from text files. We propose an improvement of the script that allows reading and writing data from / to a Database Management System such as MS SQL Server. This functionality could be implemented in MatLab through the Database Toolbox.

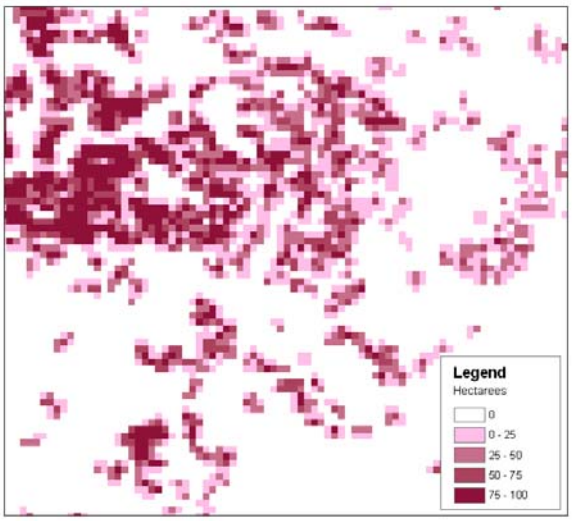
Another limitation of the smoothing methodology is the size of the resulting layers. The proposed solution of converting from float to integer values reduces notably the size of the files but produces a loss of information (see Table 2.4.1)

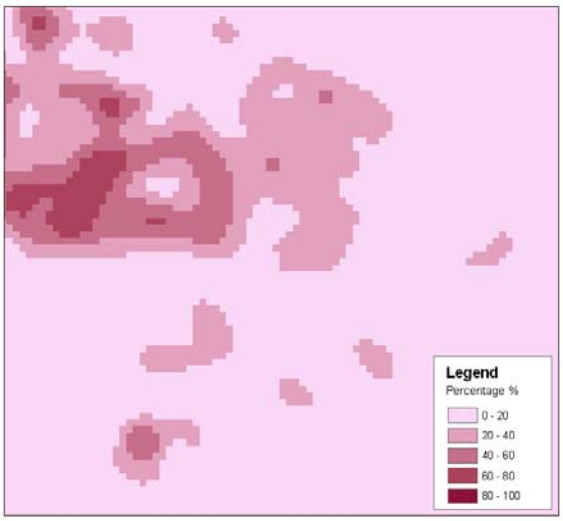
In the conceptual side we propose to smooth raw data instead of surfaces aggregated to the Reference Grid. This smoothing would work directly on Boolean values for each of the CLC classes and the results would maintain the spatial resolution of the input Land Cover layers (100 m).

**Table 3.1 — Smoothing CLC raw data** has been tested for regional scales. The results show that the only benefit of smoothing data at 100 m resolution is the precision of the final contours.



# 4 REFERENCES

**Geographic Information Management (2000)** *CORILIS Technical Report.*

**Ifen (2000)** CORILIS, Lissage de CORINE Land Cover / Methodologie CNRS - INSEE

**MATLAB** version 7.0.4.365 R14 (Service Pack 2) The MathWorks Inc, 2005

**Weber J.L., Páramo F., Breton F., Haines-Young R. (2003)** Integration of geographical and statistical data in the environmental accounting framework; methodological development based on two case studies

**EEA (2006)** Land accounts for Europe 1990-2000. Towards integrated land and ecosystem accounting. EEA Report nº 11/2006.

# 5 SCRIPTS

The following code is subject to General Public License (GPL). Please refer to European Topic Centre on Land Use and Spatial Information (etclusi@uab.es) for further information.

## 5.1. PREMATLAB.AML

```
/* PREMATLAB.AML – Preparing input data for MatLab
/* Version: February 2008

/* Argument CLC = Name of CLC coverage in grid format
&ARGS CLC
grid
/* Processing Totals: aggregating & resampling
land = aggregate(%CLC% < 255, 10, sum, expand, DATA)

/* Creating the mask to clip the data in a later step.
../OUTPUT/mask = setnull(land == 0, 1)

/* Processing 44 classes separately: aggregating & resampling
&do i := 1 &to 44
       valor%i% = aggregate(%CLC% == %i%, 10, sum, expand, DATA)
&end
q

/* Exporting to Ascii (totals and 44 values)
gridascii land land.txt
&do i := 1 &to 44
       gridascii valor%i% valor%i%.txt
&end

/* Deleting temporary grids (totals and 44 values)
kill land all
&do i := 1 &to 44
       kill valor%i% all
&end
```

## 5.2. RUN.M

```
function run()
% DataPrefix => Prefix in the source files ej. data/gis/ASCII1/VALUE_
% InputDir => Folder name for the source files
% OutputDir => Folder name for the smoothed files
% AreaFile => Path to the TOTALS file
% R => Radius in map units (resolution of input data)

DataPrefix = 'Valor_';
InputDir = 'C:\WORKSPACE\WK\INPUT\';
OutputDir = 'C:\WORKSPACE\WK\OUTPUT\';
AreaFile = 'C:\WORKSPACE\WK\INPUT\land.txt';
R = 5;
% Reading source files
DataFilesStruct = dir([InputDir DataPrefix,'*']);
% Pick only the file names
for i = 1:size(DataFilesStruct,1)
    DataFiles{i} = [InputDir DataFilesStruct(i).name];
end
% Send to CorilisFloat script
corilisFloat(R,DataFiles,AreaFile,OutputDir);
```

## 5.3. CORILISFLOAT.M

```
% initial parameters
%%%%%%%%%%%%%%%%%%%%%
% R => Smoothing radius
% AreaFile => Total area file e.g. data/gis/ASCII/TOTAL.txt
% DataFiles => Cell array with file names
% OutputDir => Output directory for smoothing results

function corilis(R, DataFiles, AreaFile, OutputDir);
%% Disable division by zero warning
warning off MATLAB:divideByZero;

%% Starting time
timestart = uint8(clock());
disp(sprintf('Smoothing started at
%d:%d:%d',timestart(4),timestart(5),timestart(6)));

%% Reading header information
f = fopen(AreaFile);
line = fgetl(f);
TotalCols = str2num(line(7:size(line,2))); % Number of columns
line = fgetl(f);
TotalRows = str2num(line(7:size(line,2))); % Number of rows
% Skip xllcorner, yllcorner, cellsize and ndata_value
for i=1:4
    fgetl(f);
end
% Reading Total area values
ftotales = uint8(fscanf(f, '%d',[TotalCols,Inf]));
ftotales = ftotales';
fclose(f);
% Create convolution window of radius R
win = single(createWindow(R));
% Smoothing Total Area file and clearing
stotales = single(conv2(single(ftotales),win,'same'));
clear ftotales;
% For each file:
for i = 1:size(DataFiles,2)
    % Open source file
    f = fopen(DataFiles{i});
    % Open target file
```

```
    [path,filename,extension] = fileparts(DataFiles{i});
    fo = fopen([OutputDir filename 'S' extension],'wt');
    % skip first 4 header lines and print them in the output file
    for j=1:5
        fprintf(fo,'%s\n',fgetl(f));
    end
    fgetl(f); % skip last header line
    fprintf(fo,'%s\n','nodata_value -1.00000'); % print last header line with
nodata flag
    fdata = uint8(fscanf(f, '%d',[TotalCols,Inf])); % read source data
    fdata = fdata'; % transpose matrix
    fclose(f); % close source file
    sdata = single(conv2(single(fdata),win,'same')); % perform convolution analysis
    clear fdata; % clear data
    % Divide results by total area smoothed.
    % If a division by zero id done a Nan value is stored: 0/0 = NaN, N/0 = Inf
    % datasuavizada = sdata %(sdata ./ stotales);
    datasuavizada = (sdata ./ stotales);
    clear sdata % clear sdata
    datasuavizada(isnan(datasuavizada)) = -1; % Replace NaN by -1
    datasuavizada = datasuavizada'; % Transpose the matrix again
    % datasuavizada = int8(datasuavizada'); % Converting to integer (old
    % system)
    % formatting output values using one place for the entire part and 4
    % places for decimals.
    format = ' %1.5f';
    for j=2:TotalCols
        format = [format ' %1.5f'];
    end
    format = [format '\n'];
    % Writing smoothed data in the output file
    fprintf(fo,format, datasuavizada);
    fclose(fo);
    timenow = uint8(clock());
    disp(sprintf('Smoothing of file %s completed successfully at
%d:%d:%d',DataFiles{i},timenow(4),timenow(5),timenow(6)));
end

%% Finishing time
timeend = uint8(clock());
disp(sprintf('Smoothing completed at %d:%d:%d',timeend(4),timeend(5),timeend(6)));
```

## 5.4. CREATEWINDOW.M

```
% Radius => if Radius = 5 it creates a 9x9 window, etc

function window = createWindow(Radius)
for i = 1:Radius * 2 - 1
    for j = 1:Radius * 2 - 1
        dist = sqrt((radio - i)^2 + (Radius -j)^2);
        %distance([Radius,Radius],[i,j]);
        if dist <= radio
            window(i,j) = (1 - (dist/Radius)^2)^2;
        end
    end
end
```

## 5.5. POSTMATLAB.AML

```
/* POSTMATLAB.AML – Importing MatLab outputs.
/* Version: February 2008


&do i := 1 &to 44
       asciigrid valor%i%s.txt valor%i% float
&end


GRID
suma =
SUM(valor1,valor2,valor3,valor4,valor5,valor6,valor7,valor8,valor9,valor10,valor11,
valor12,valor13,valor14,valor15,valor16,valor17,valor19,valor18,valor20,valor21,val
or22,valor23,valor24,valor25,valor29,valor26,valor27,valor28,valor30,valor31,valor3
2,valor33,valor34,valor35,valor36,valor37,valor38,valor39,valor40,valor41,valor42,v
alor43,valor44)


&do i := 1 &to 44
       tmp%i% = (valor%i% / suma) * 100
       int_sm%i% = int(tmp%i%)
       r5sm%i% = int_sm%i% * mask
&end


/* Deleting temporary grids
&do i := 1 &to 44
       kill valor%i% all
       kill tmp%i% all
       kill int_sm%i% all
&end
Q
```