

The Lumileds computer program

Citation for published version (APA):

Spoelstra, J. (1997). *The Lumileds computer program*. (IWDE report; Vol. 9706). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1997

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Rapport IWDE 97 - 06

The Lumileds computer program

J. Spoelstra

december 1997

Instituut
Wiskundige Dienstverlening
Eindhoven

Report IWDE 97-06

The Lumileds computer program

J Spoelstra

wskjs@puknet.puk.ac.za

December 18, 1997

Contents

I	User's Guide	2
1	Introduction	2
2	Starting the program	3
3	Editing an M-file	4
4	Constructing the Minor road block	4
5	Defining a road section	6
6	Reading the IES-table of a luminaire	7
7	Analysing a road block or given luminaire	8
8	Optimizing	9
9	Saving an IES-file	10
II	Program-listings	12
10	Initialisation Programs	12
11	Programs for the objective function	24
12	Programs for the optimizing	26
13	Programs for the road measurements	30
14	Program for IES files	36
15	Utilities	37

Part I

User's Guide

1 Introduction

This program runs in a MATLAB-environment, and can be used for the following purposes:

1. To **construct** a *minor road block* from:
 - a given set of possible beam and optical element types (in the form of sets of IES-files),
 - a given list of deflection angles (each of which corresponds to a selection of one of the IES-files),
 - a given list of placements of the optical elements on the block (in the form of rotation angles),
 - a given list of selections of beam and optical element types from the IES-files (*i.e.*, of a beam type for the already specified deflection angle),
 - a given tilt of the complete block,
 - a given rotation of the complete block, and
 - if so desired, adding a mirror-symmetrical counterpart as part of each block.
2. To **analyse** the constructed road block,
 - by viewing a graphical representation,
 - by computing its effect on a user-defined road section,
 - by viewing its effect on the user defined road section.
3. To **read in and analyse** a luminaire (given in the form of an IES-table).
4. To determine an **optimal** road block by selecting the best values for the variables from those possible in the above list,

- optimal in the sense of closest correspondance to a given IES-table, or
 - optimal in the sense of attaining a given set of criteria for the quality of the road lighting, and, if attainable, achieving it with as low an output as possible.
5. To compute and **save to disk** an IES-file for such a minor road block for the given setup, or as determined by the optimisation.

2 Starting the program

The program writes intermediate and final results in the following directory: `c:\matlab\results`, so that the user must create this directory before starting. Please ensure that the directory where the program will find the files `target.m` and `inr16.m` is set correctly in the M-file `GETTARGT.M` — change it if necessary.

To start the program the MATLAB program must first be started. This should be done by double-clicking the `Matlab`-shortcut icon. The Properties of the shortcut must be set in such a way that the program starts in the directory

```
c:\Matlab\leds\ton
```

Once in the correct directory, the `path` must be set correctly. This can be done automatically by running the M-file `lumileds.m`. This is achieved by typing, after the `>>`-prompt, `lumileds` and the `ENTER`-key:

```
>> lumileds
```

This program sets up the correct path, and asks that the user edit a few programs to specify his questions.

When the files `MRBDATA.M` and `ROADDATA.M` have been edited, the minor road block is constructed, and the information is set up, by the program `MRBINIT.M`:

```
>> mrbinit
```

Before restarting with a different road, clear the memory by typing and entering

```
>> clear
>> clear global
```

3 Editing an M-file

This can be achieved in the MATLAB environment by

1. clicking **File**,
2. clicking **Open M-file**,
3. selecting the file from the list, and
4. double clicking the file name

The file is then opened in Notepad and must be saved before any changes will take effect.

4 Constructing the Minor road block

All information for the given setup must be supplied in the M-file `MRBDATA.M`. The following listing gives the options and serves as example:

```
c:\matlab\leds\ton\roaddata.m
-----
% MRBDATA

N_sources = 36;           % The number of sources (singles|pairs|triplets|quads)
                        % in the roadblock?

pash=0
inputflux = 72000        % "pash=1" indicates that files are in Mike Pashley-format
                        % If the inputflux is known, this must be to that value,
                        % else set it to 0

numberoffilesets = 1;   % IF spreading or some other extra dimensionality is
                        % not available, NUMBEROFFILESETS must be 1.
                        % Else, set NUMBEROFFILESETS to the number of series of
                        % files available, AND
                        % set "fullname2", "fullname3", ...
```

```

% to the basic names of the other sets of files

deflctoflastfile = 46; % The deflection of the last of the files in the set.
% The number of files in the set is deduced from this
% by the formula
% (deflctoflastfile/2+1)

fullname1 = 'c:\leddata\g2g2\g2g200'
%fullname2 = 'c:\leddata\t1g2\t1g202'
%fullname3 = 'c:\leddata\t1g2\t1g203'
% The name of the directory and first part of the
% name of the file containing the source information.
% This info must be available in the named
% " drive:\directory\filename00.ies "
% " drive:\directory\filename02.ies "
% " drive:\directory\filename04.ies"
% . . .
% " drive:\directory\filename10.ies "
% . . .
% " drive:\directory\filename48.ies "
% Specify only the first part, and with no extension!

symmetrical = 1; % Setting this to any other value than 1 would construct
% a road block without a mirror image block

%a=randn(1,N_sources); % Activating this would cause a random distribution
%a=a-min(a); % in output of the source to take effect
%quality=1.333*(a/max(a))+0.333; % varying NORMALLY between 33% and 166% of normal output

tiltperk = 60.00 ; % The maximum tilt allowed. (degrees)
rotperk = 90.00 ; % the maximum rotation allowed (degrees)
prismperk = 46 ; % The maximum allowed prism deflection (degrees)

% The TILT (or a starting value for the tilt) (degrees)
% The ROT (or a starting value for the rotation) (degrees)
% The set-up to use, in the format
% prism deflection, prism-placement, other-factor

tilt = 43.18 ;
rot = 27.56 ;
setup = [...
45.83 6.75 .00
43.61 -101.30 .00
38.18 -72.07 .00
41.39 -139.36 .00
37.45 -29.22 .00
45.34 -162.45 .00
44.37 153.57 .00
45.39 124.64 .00
41.61 138.01 .00
40.71 -40.13 .00
38.49 105.76 .00
35.26 174.48 .00
34.74 94.79 .00
23.76 17.01 .00
28.14 41.70 .00
27.47 32.25 .00

```

```

33.59 -27.15 .00
26.00 -47.84 .00
28.49 75.14 .00
29.44 -172.17 .00
23.64 -91.59 .00
24.76 74.10 .00
25.86 -21.98 .00
24.27 117.26 .00
23.06 94.60 .00
24.34 -14.53 .00
18.85 43.04 .00
18.51 145.58 .00
16.76 -174.67 .00
16.17 -.77 .00
15.52 -68.14 .00
12.75 -132.37 .00
12.47 53.65 .00
12.93 103.47 .00
12.66 -20.83 .00
.44 -99.06 .00];

```

5 Defining a road section

All information needed to specify the dimensions of the road and the lighting setup must be specified in the file ROADDATA.M.

```

c:\matlab\leds\ton\roaddata.m
-----
% PLEASE ENTER ALL DESIRED VALUES IN THE FOLLOWING LIST
clear global newroad
global newroad
newroad = 1;

K1 = 2.5 ; % Left kerb
LW1 = 3.5 ; % Left lane
LW2 = 3.5 ; % Right lane
K2 = 2.5 ; % Right kerb

x_displ_obs = (57.3 + 171.9 )/2 ; % Distance of observers from the centre of
% the observed area

%x_displ_obs = 81

% ----- Luminaire-setup -----
F_height = 7 ; % Height of luminaires
t = 42; % Temporary value to define the pole-coordinates easier

%Poles =[-4*t 0
% -3*t 0
% -2*t 0
Poles=[ -1*t 0
0*t 0
t 0
2*t 0] % The number of poles adjusted automatically
% to the number here

```

```

                                % ----- Road type data -----
rtafile = 'c:\matlab\leds\rtab2.txt' ;
                                % Full Name of the file in which the
                                % reflectance-table is given:

                                % ----- Essential target values for the --
                                % ----- illumination of the road. -----

                                % These must be ESSENTIAL. The program will
                                % first try to achieve these, before considering
                                % any other optimizing effectively

want_t_eta = 0.9 ;                % Desired Total Efficacy, between 0 and 1.
                                % If not important, set to 0
want_r_eta = 0.7 ;                % Desired Road Efficacy, between 0 and 1.
                                % If not important, set to 0
want_lum    = 1 ;                 % Desired average luminance in direction of
                                % worst observer. The lumens will be scaled so
                                % as to always achieve this.
want_o_uni  = 0.43 ;             % Desired overall uniformity for the worst of
                                % two observers, between 0 and 1.
                                % If not important, set to 0.
want_l_uni  = 0.60 ;            % Desired lengthwise uniformity for the worst
                                % of 10 observers, between 0 and 1.
                                % If not important, set to 0.
want_TI     = 10 ;               % Desired Threshold Increment, as a percentage.
                                % If not important, set to 100.
want_SR     = 0.30 ;            % Desired Surround Ratio, between 0 and 1.
                                % If not important, set to 0.

```

6 Reading the IES-table of a luminaire

If an existing luminaire has to be analysed, it must first be read in to memory. This is done by the program LEESIES.M. The program returns the IES-table and its flux as its response, and must be given the path to the file, and the name of the file, in the following form:

```
>> [mylum, myflux] = leesies('c:\leddata\t1g1\t1g1203')
```

The variable `mylum` will henceforth contain the IES-table of the luminaire, and can be analysed in the same way as a constructed road block.

If, by viewing the file with the command `rond(mylum)` it is found that it needs to be mirrored in the 0-180 degrees plane, type in the following instructions (where the name: `mylum` is the name the user has chosen for the luminaire information):

```
>> mylum = flipud(mylum);  
>> mylum = frotalf(mylum,5);
```

This last program `frotalf.m`, can be used for any rotation of the file through an angle (given in degrees).

7 Analysing a road block or given luminaire

The (condensed) IES-table for the constructed road block is stored in the memory of the computer under the name `MRB`. The IES-table of any intermediate road block constructed by the optimizing program will be available under the name `SS`.

To view a graphical representation of an IES-table with the name `TABLE`, use the program `ROND.M`:

```
>> rond(TABLE);
```

This will create a two-dimensional image of the information. To view it three-dimensionally, type

```
>> view(3)
```

or use any other of the conventions for viewing angle, as can be found by typing

```
>> help view
```

To compute the effect on the road (as defined in `ROADDATA.M`), the function-program `ROADMEAS.M` must be used. This program needs three arguments: the name of the IES-table, the *input flux* in the IES-table (for an existing IES-table, this can be taken as its flux), and a key indicating what needs to be computed. The conventions for this key is as follows:

```
key = 1 : full data to screen  
key = 2 : full data to filename "rM_D_T.txt", with M,D,T  
          one or two digit numbers containing the month,
```

```
the day and the hour of file creation
key = 4 : draw picture of road illumination
key = 8 : draw contour maps of road illumination
key = 16: Write the luminances in the direction of the
different observers to a file
Add values of key, if more than one action is wanted.
```

Therefore, to see the data on the screen, and draw an picture of the road illumination, enter the following command for a multisource road block with IES table called MRB, and an inputflux of 72000:

```
>> roadmeas(MRB, 72000, 5);
```

and to write all luminances to a file, enter the following:

```
>> roadmeas(MRB, 72000, 16);
```

If it is an existing IES-file that has been read in, and that has to be analyzed, add the name of the road block to the list of arguments:

```
>> roadmeas(MRB, mrbflux, 5, 'my eie IES l^eer');
```

where the variable `mrbflux` can be obtained by entering

```
>> mrbflux = fluxies(C_list, gam_list,MRB);
```

and the program will suppress the road-block information, and print the sentence `my eie IES l^eer` at the top of the block.

8 Optimizing

To optimize the adjustment and selection of the optical elements in a road block, the strategy must be selected. In the following listing a few standard strategies are given.

The user should delete the “%” sign in front of the strategy that he finds applicable, or define his own, or copy one such strategy to the last line without the “%” sign.

```

c:\matlab\leds\ton\optidata.m
-----
% "STRATEGY" contains values that the variable "STAGE" will take on at different runs
% of the basic optimization routine.

% The values are constructed as follows:

% Add to stage=0 the following:
%      1   to optimize over deflection angle
%      2   to optimize over placement angle (= rotation of prism)
%      4   to optimize over third factor
%      8   to optimize over tilt
%     16   to optimize over rot
%     32   to aim at the target distribution, else only road-measures
%     64   to group over deflection angles
%    128   to group over placement angle
%          Grouping over third factor is at present done automatically to
%          the discrete files

% If you do not want to change the deflection angles
%strategy = [ 26 2]
% Case B: if the initial data is very bad
%strategy = [ 58 26 2]
% If you want everything except for third factor
%strategy = [ 26 27 1 2 3 2]
% Case B: if the initial data is very bad
%strategy = [ 58 26 27 1 2 3 2]

% If you want to include a third factor
%strategy = [ 4 31 7 6 5 4 3 2]
% Case B: if the initial data is very bad
%strategy = [ 63 4 31 7 6 5 4 3 2]

% If you want to optimise over everything and then group
% over deflection angles
%strategy = [ 4 31 7 6 5 4 3 2 73 6 65 2]

strategy=[ 67 3 65 131]

%          IF you want to group, then the variable WANTGROUPSIZE must be
%          set
wantgroupsize = [12 N_sources N_sources]

```

9 Saving an IES-file

If the information in a constructed IES-table needs to be saved to disk as an IES-file, it can be done by the function-program `VORMIES.M`. This function needs two arguments: the name of the IES-table, and the name under which it must be saved:

```
>> vormies(SS, 'c:\matlab\results\jaapsp.ies');
```

This file should conform in most respects to the IES-specifications.


```

% It uses data concerning the road as given in the
% file  ROADDATA.M ,
% concerning the sources as given in the
% file  SORSDATA.M
% and concerning the road block as given in the
% file  MRBDATA.M

% A basic set of information needs to be available for
% all applications. These quantities are declared as
% "GLOBAL" variables.

global K1          LW1          LW2          K2          ...
        F_height    S_poles    x_displ_obs  N_grid    Poles

global rtab        beta_list    tangam_list

global want_r_eta  want_t_eta  want_SR      want_lum    ...
        want_o_uni  want_l_uni  want_TI

global newroad     target_MRB   tonvh        symmetrical  quality SS

global SORS_IES    C_list       gam_list     SORS_FLX ...
        fullname1   fullname2   fullname3   fullname4   fullname5

global tilt        rot          setup        fullname ...
        tiltperk    rotperk    prismperk   inputflux ...
        facperk     ...
        MRB         MRB_outflux N_sources ...
        HaRB        HaRB_influx HaRB_outflux

% The following file reads information as to the type of road, its
% measurements and the observers.

roaddata; %-----

% The following M-file reads the reflectance-table for the road type
% as given in the file roaddata.m

leesrtb; %-----

% The following files reads the information as to the specific road block:
% the type of sources, the deflection optics, the placement and a third factor,
% if present, and constructs the IES-file for the combined Road Block consisting
% of the number of sources as specified in mrbdata

mrbdata; %-----

facperk = numberoffilesets - 1;
if (exist('quality') == 0 )
    quality = ones(1,N_sources);
elseif size(quality,2) ~= N_sources
    disp('Redefining quality-vector due to incorrect length');
    quality = ones(1,N_sources)
end

% The following file reads the information as to the types of

```

```

% sources and optics under consideration.

if (exist('SORS_IES') == 0)
    getsors;
else
    disp('Must I read and set up all the source data?');
    antwoord = input('("N|n" => no. Anything else => yes) ', 's');
    if antwoord ~= 'n'
        getsors; %-----
    end
end

% The following program constructs the IES-file for the half road block
% (before tilting, rotating and adding mirror image).

[HaRB, HaRB_influx, HaRB_outflux] = make_hrb(setup); %-----

% The following function forms the IES-table for the combined Road Block
% consisting of half-road-block given as input, tilting it, rotating it and
% adding the mirror image.

[MRB, MRB_outflux] = total_rb(tilt, rot, HaRB); %-----

% The following function reads TARGET-IES-tables.
% First a target for the half Road Block, target_HaRB
% as well as a target for the complete MRB, called, target_MRB.

gettarget; %-----

SS = MRB;

disp(' I now have all road data, source data, mrb-data, targets');
clear
pack

c:\matlab\leds\ton\leesrtb.m
-----
fid = fopen(rtabfile,'rt');
a = fscanf( fid, '%d', 1 );
tangam_list = fscanf( fid, '%f ', a);
b = fscanf( fid, '%d', 1 );
beta_list = fscanf( fid, '%d ', b );
c = fscanf( fid, '%f', 1 );
for i = 1:b
    bb = fscanf( fid, '%d ', a );
    rtab(i,1:a) = bb';
end;
rtab = c*rtab;
fclose(fid);
clear a b c bb;

c:\matlab\leds\ton\getsors.m
-----

```

```

disp(['Starting to read the data files named ' fullname]);

SORS_IES = zeros(25*37,31*numberoffilesets);
SORS_FLX = zeros(25,numberoffilesets);
indexset = [1:31] ;
tic
for jj=1:numberoffilesets
    eval(['fullname = fullname' int2str(jj)])

leestyp =['leesmike'
          'leespash'];

for i = 0:2:8
    disp([num2str(i) setstr(176)]);
    indekse = round(i/2)*37 + [1:37];
    stringie = [fullname '0' num2str(i) '.ies'];
    eval(['[ies, flux] = ' leestyp(pash+1,:) '(stringie);'])
    SORS_IES(indekse, (jj-1)*31+indexset) = round(ies);
    SORS_FLX(round(i/2)+1, jj) = flux;
end;

for i = 10:2:deftctoastfile
    disp([num2str(i) setstr(176)]);
    indekse = round(i/2)*37 + [1:37];
    stringie = [fullname num2str(i) '.ies'];
    eval(['[ies, flux] = ' leestyp(pash+1,:) '(stringie);'])
    SORS_IES(indekse, (jj-1)*31+indexset) = round(ies);
    SORS_FLX(round(i/2)+1, jj) = flux;
end;
end

disp(['..... It took ' num2str(toc) ' seconds for all']);

C_list = [0:5:355]';
gam_list = [0:3:90];

c:\matlab\leds\ton\make_hrb.m
-----
function [HaRB, HaRB_influx, HaRB_outflux] = make_hrb(setup)
global C_list gam_list SORS_IES SORS_FLX quality

nums = size(SORS_FLX,2)-1;
HaRB = zeros( 72 , 31);
HaRB_outflux = 0;

for i = 1:(size(setup,1))
    prism = setup(i,1);
    alpha = setup(i,2);
    gammi = setup(i,3);
    if nums > 0
        if gammi >= nums
            gammi=nums - 0.00001;
        end
    end
    end
    indic = fix(gammi);

```

```

indics= indic*31 + [1:31];
indx = floor(prism/2) ;
n_on = indx*37 + [1:37];
n_bo = n_on + 37 ;
t = rem(prism,2)/2 ;

I_k = SORS_IES(n_on, indics);
I_g = SORS_IES(n_bo, indics);
ss = (1-t)*I_k + t*I_g ;
f_k = SORS_FLX(1+indx ,indic+1);
f_g = SORS_FLX(1+indx+1,indic+1);
fm = (1-t)*f_k + t*f_g ;

if nums >0
    indic = indic+1;
    indics= indics+31;
    I2_k = SORS_IES(n_on, indics);
    I2_g = SORS_IES(n_bo, indics);
    ss2 = (1-t)*I2_k + t*I2_g ;
    f2_k = SORS_FLX(1+indx ,indic+1);
    f2_g = SORS_FLX(1+indx+1,indic+1);
    f2m = (1-t)*f2_k + t*f2_g ;

    t = rem(gammi,1);
    ss = (1-t)*ss + t*ss2;
    fm = (1-t)*fm + t*f2m;
end

s1 = flipud(ss);
s1 = s1(2:(size(s1,1)-1),:);
ss = [ss
      s1];
sss = frotalf(ss, alpha);
if quality(i) ~= 1
    sss = quality(i)*sss;
    fm = quality(i)*fm;
end;
HaRB = HaRB + sss ;
HaRB_outflux = HaRB_outflux + fm ;
end;

HaRB_influx = 1000*size(setup,1);

c:\matlab\leds\ton\total_rb.m
-----
function [MRB, MRB_outflux] = total_rb(tilt, rot, HaRB);
global C_list gam_list symmetrical
% Tilt and rotate
[C_hat,gam_hat] = intrpris(tilt,C_list,gam_list);

tempC = [C_list' 360];
temptans = [HaRB
            HaRB(1,:)];
tempg = [gam_list 180];
temptans = [temptans zeros(size(temptans(:,1)))];

```

```
SS = interp2( tempC, tempg', temptans', C_hat, gam_hat);
SS1 = frotalf( SS, rot);
```

```
if ( symmetrical == 1 )
    % Add contribution of mirror image block
    nn1 = [37:-1:1 72:-1:38];
    SS2(:, :) = SS1(nn1, :);
    %Final
    MRB = SS1 + SS2;
    MRB_outflux = fluxies(C_list, gam_list, SS);
else
    MRB = SS1;
end
```

```
c:\matlab\leds\ton\gettarget.m
```

```
-----
tempC = [0:6:360];
tempg = [0:3:78 90 180];
```

```
fid = fopen('c:\leddata\iesdata\target.m', 'rt');
idflux = fscanf( fid, '%f', 1);
a = fscanf( fid, '%d', 2 );
C_l = fscanf( fid, '%d ', a(1));
gam_l = fscanf( fid, '%d ', a(2))';
for i = 1:a(1)
    bb = fscanf( fid, '%d ', a(2) );
    ST(i, 1:a(2)) = bb';
end;
fclose(fid);
temptans = [ST
            ST(1, :)];
temptans = [temptans zeros(size(temptans(:, 1)))];
target_MRB = round(interp2( tempC, tempg', temptans', C_list, gam_list))';
target_MRB(:, 29:31) = zeros(72, 3);
```

```
fid = fopen('c:\leddata\iesdata\idealies.m', 'rt');
tonflux = fscanf( fid, '%f', 1);
a = fscanf( fid, '%d', 2 );
C_l = fscanf( fid, '%d ', a(1));
gam_l = fscanf( fid, '%d ', a(2))';
for i = 1:a(1)
    bb = fscanf( fid, '%d ', a(2) );
    ST(i, 1:a(2)) = bb';
end;
fclose(fid);
temptans = [ST
            ST(1, :)];
temptans = [temptans zeros(size(temptans(:, 1)))];
tonvh = round(interp2( tempC, tempg', temptans', C_list, gam_list))';
tonvh(:, 29:31) = zeros(72, 3);
clear global ST
```

```
c:\matlab\leds\ton\leesmike.m
```

```
-----
function [ies, flux] = leesmike(stringie)
```

```

[iess,flux] = leesies(stringie);
ies(1:37,1:31) = iess(1:37,1:31);

c:\matlab\leds\ton\leespash.m
-----
function [ies,flux] = leespash(stringie)
disp(stringie);
skrywer= ' ';
fid = fopen(stringie,'rt');
line = 'abcde';
kar = '';
while any(line ~= 'NONE'), %feof(fid)==0
    kar = fscanf(fid,'%c',1);
    line = [ line(2:5) kar];
end;
disp(skrywer)
g = fscanf(fid,'%f',3);
n_gam = fscanf(fid,'%f',1);
n_C = fscanf(fid,'%f',1);
g = fscanf(fid,'%f',5);
g = fscanf(fid,'%f',3);
a = fscanf(fid,'%f',n_gam);
gamma_list = a;
a = fscanf(fid,'%f',n_C);
Cc_list = a;
for i = 1:n_C
    a = fscanf(fid,'%f',n_gam);
    ies(i, :) = a';
end
fclose(fid);

ss = (ies(2:360,2:90) + ies(2:360,3:91) + ies(1:359,2:90) + ies(1:359,3:91) )/4;

eerstery = (ies(360,2:90) + ies(360,3:91) + ies(1,2:90) + ies(1,3:91) )/4;
eerstekolom = ies(:,1);
laastekolom = ies(:,91)/2; % Eintlik ( .. + 0)/2;

ies = [eerstekolom [eerstery ; ss ] laastekolom];

s1 = ies(2:360,:);
s1 = (s1 + flipud(s1))/2;

ies = [ies(1,:)
      s1];

flux = fluxies([0:359], [0:90], ies )

iess(1:360,1:31) = ies(1:360,[0:3:90]+1 );
clear ies;
ies(1:37, 1:31) = iess([0:5:180]+1,1:31);

ies = round(ies);

```

```

c:\matlab\leds\ton\fluxies.m
-----
function lig = fluxies(C_list, gamma_list, i_tab);
% FLUXIES computes the surface integral for an IES-table
% by a combined formula, the formula to be given later
% with the (n+1)st C taken as 360

global fluxes
n = length(C_list);
np1 = n+1;
gamma_list = gamma_list/180*pi;
deltaC = (C_list(2) - C_list(1))*pi/180;
i_tab(np1,:) = i_tab(1,:);
m = length(gamma_list);
mm1 = m-1 ;
deltag = gamma_list(2:m) - gamma_list(1:mm1);
lig1 = (sum( ...
sum( (i_tab(1:n,1:mm1) + i_tab(2:np1,1:mm1)) .* ...
( sin(gamma_list(1:mm1)).*deltag ) ...
+ ...
sum( ...
sum( (i_tab(1:n, 2:m) + i_tab(2:np1, 2:m )) .* ...
( sin(gamma_list(2:m) ).*deltag ) ) /4*deltaC;

mat = i_tab(1:n,1:mm1) + i_tab(2:np1,1:mm1) ...
+ i_tab(1:n, 2:m) + i_tab(2:np1, 2:m) ;
deel2 = cos(gamma_list(1:mm1)) - cos(gamma_list(2:m));

lig2 = sum( sum(mat) .* deel2 ) /4*deltaC;

lig = (lig1+lig2)/2;
return;

c:\matlab\leds\ton\leesies.m
-----
function [ies,flux] = leesies(stringie)

fid = fopen(stringie,'rt');
line = 'abcde';
kar = '';
while any(line ~= 'NONE'), %feof(fid)==0
kar = fscanf(fid,'%c',1);
line = [ line(2:5) kar];
end;

g = fscanf(fid,'%f',3);
lum_lam = g(2);
multiplier = g(3);
n_gam = fscanf(fid,'%f',1);
n_C = fscanf(fid,'%f',1);
g = fscanf(fid,'%f',5);
if g(1) ~= 1
disp(' I am sorry --- I only handle photometric types C -- ref IES LM-63-1991, 4.17');
fclose(fid);
return;
end;

```

```

g      = fscanf(fid,'%f',3);
a      = fscanf(fid,'%f',n_gam);
gamma_list = a;
a      = fscanf(fid,'%f',n_C);
Cc_list = a;
for i = 1:n_C
    a = fscanf(fid,'%f',n_gam);
    ies(i , :) = a';
end
fclose(fid);

if (multiplier ~= 1.0 )
    disp('multiplier not 1 ..... multiplying (just) the IES-file by it!!!');
    ies = ies*multiplier;
end

%laastegam=gamma_list(n_gam)

if gamma_list(1)==0.0
    if gamma_list(n_gam) == 90
        tempg = [gamma_list' 180];
        temptans = [ies zeros(size(ies(:,1)))];
    elseif gamma_list(n_gam) == 180
        tempg = gamma_list';
        temptans = ies;
    else
        disp(' The final vertical angle not valid according to 4.17 of IES LM-63-1991');
        return
    end
elseif gamma_list(1)==90.0
    if gamma_list(n_gam) == 90
        disp(' From 90 degrees to 90 degrees for vertical angles --- are you joking?');
        return
    elseif gamma_list(n_gam) == 180
        tempg = [0 gamma_list'];
        temptans = [zeros(size(ies(:,1))) ies];
    else
        disp(' The final vertical angle not valid according to 4.17 of IES LM-63-1991');
        return
    end
else
    disp(' The first vertical angle not valid according to 4.17 of IES LM-63-1991');
    return
end

ies = temptans;

if Cc_list(1)==0.0
    if Cc_list(n_C) == 0
        tempC = [0:5:360];
        temptans = ies;
        for i=tempC(2:73)
            temptans = [temptans
                        ies];
        end
    elseif Cc_list(n_C) == 90
        tempC = [Cc_list

```

```

        Cc_list(2:n_C)+90
        Cc_list(2:n_C)+180
        Cc_list(2:n_C)+270];
    s1 = flipud(ies);
    s1 = s1(2:(size(s1,1)),:);
    ies = [ies
           s1];
    s1 = flipud(ies);
    s1 = s1(2:(size(s1,1)),:);
    temptans = [ies
                s1];
elseif Cc_list(n_C) == 180
    tempC = [Cc_list
            Cc_list(2:n_C)+180];
    s1 = flipud(ies);
    s1 = s1(2:(size(s1,1)),:);
    temptans = [ies
                s1];
elseif (Cc_list(n_C) > 180)
    while (Cc_list(n_C) < 360)
        step = (Cc_list(n_C)-Cc_list(1))/n_C;
        Cc_list = [Cc_list
                  Cc_list(n_C,1)+step];
        ies = [ies
              zeros(size(ies,2))];
        n_C = n_C+1;
    end
    tempC = Cc_list ;
    temptans = ies;
else
    keyboard
    disp(' A: The final horizontal angle not valid to 4.18 of IES LM-63-1991');
    return
end
elseif ( Cc_list(1) == 90.0 )
    if ( Cc_list(n_C) == 270 )
        tempC = [Cc_list
                Cc_list(2:n_C)+180];
        s1 = flipud(ies);
        s1 = s1(2:(size(s1,1)),:);
        temptans = [ies
                    s1];
        tempC = tempC-90;
        halfn_C = fix(n_C/2);
        indekse = [ (n_C+halfn_C):(n_C+n_C-1-1) (1:(n_C+halfn_C))] ;
        temptans = temptans(indekse,:);
    else
        disp(' The final horizontal angle not valid according to 4.17 of IES LM-63-1991');
        return
    end
else
    disp(' B: The first horizontal angle not valid according to 4.17 of IES LM-63-1991');
    return
end
ies = round(interp2( tempC, tempg', temptans', [0:5:355] , [0:3:90] ))';
flux = fluxies([0:5:355], [0:3:90], ies);

```

```

disp(['Computed flux = ' num2str(flux) ', and read flux = ' num2str(lum_lam) ]);
if ( (flux-lum_lam)/(lum_lam+flux)*200 > 9 )
    disp(' The flux read in and the computed flux differs by more than 9%');
    disp(' I will take the computed flux as the valid value!')
else
    flux = lum_lam;
end;

```

c:\matlab\leds\ton\intpris.m

```

-----
function [C_hat,gam_hat] = intpris(delta,C_list,gam_list)
% INTPRIS determines the (C,gam)-coordinates that corresponds
% to an IES-table IT, tilted upwards in the plane C=0 through
% an angle DELTA (in degrees), positive if tilted upwards.
% All data is given in degrees.
r_d = pi/180;
    % DIVIDE radians by this to get DEGREES
delta = delta*r_d;
sind = sin(delta);
cosd = cos(delta);
%NOT NOW!:      % work with half the data to save time,and mirror it.
n = length(C_list) ; %floor(length(C_list)/2)+1;
C      = C_list(1:n) * r_d ;
gamma  = gam_list      * r_d ;
sing   = sin(gamma);
cosg   = cos(gamma);
cosC   = cos(C);
cosCsing = cosC*sing;
sinCsing = sin(C)*sing;

gam_hat= acos( (cosd*ones(size(C)))*cosg + sind*cosCsing)/r_d;

C_hat  = atan2(          sinCsing ,          ...
               cosd*cosCsing - (sind*ones(size(C)))*cosg)/r_d;

    % Add 360 degrees where C_hat is negative
C_hat = C_hat + 359.99999*(C_hat<0); % + 0.0001*(C_hat==0);
return

```

c:\matlab\leds\ton\frotalf.m

```

-----
function i_nuut = frotalf( i_tab, alpha);
% FROTALF frotalf( i_tab, alpha) rotates a
% given source, i_tab, through "alpha"
% C_list is the list of C-values as used in the IES-table I_TAB
% gamma_list is the similar list of gamma-values
% I_TAB is the input IES-table.

global C_list;

while alpha>=360
    alpha=alpha-360;
end
while alpha<0

```

```

    alpha = alpha+360;
end

C_t      = C_list;
n        = length(C_t);
C_t(n+1) = 360;

a        = find( C_t > alpha );
i_ndex  = a(1)-1;
C_k      = C_t(i_ndex );
i_ndexp1 = i_ndex+1;
C_g      = C_t(i_ndexp1);
ind_on   = [ ((n - i_ndex  + 2):n) (1 : (n - i_ndex  + 1) ) ];
ind_op   = [ ((n - i_ndexp1 + 2):n) (1 : (n - i_ndexp1 + 1) ) ];
t        = ( alpha - C_k ) / ( C_g - C_k );
e_t      = 1.0-t;

i_nuut = e_t * i_tab(ind_on , : ) + t * i_tab(ind_op , : );

```

The above programs use an internal program called INTERP2.M, which does the same (but faster) as the following program:

```

c:\matlab\leds\ton\illoutve.m
-----
function lig = illoutve(C_list,gamma_list,i_tab,c,gam);
% ILLOUTVE computes the intensities for an IES-table for
% (C,gamma)-values between those in the given table, I_TAB.
% C_list is the list of C-values as used in I_TAB
% gamma_list is the similar list of gamma-values
% I_TAB is the input IES-table for a measured source
% C is an input matrix of C-values
% GAM is an input-matrix of gamma-values

n = length(C_list);
C_list(n+1) = 360;
i_tab(n+1,:) = i_tab(1,:);
m = length(gamma_list);
nn=size(c,1);
mm=size(c,2);
for i = 1:nn
    for j = 1:mm
        if ( gam(i,j) >= max(gamma_list) )
            lig(i,j) = 0.0;
        else
            a = find(C_list>c(i,j));
            i_ndex = a(1)-1;
            i_ndexp1 = i_ndex+1;

            a = find(gamma_list>gam(i,j));
            j_ndex = a(1)-1;
            j_ndexp1 = j_ndex+1;

            t = (c(i,j) - C_list(i_ndex)) / ...
                (C_list(i_ndexp1) - C_list(i_ndex));

```

```

    e_t = 1.0-t;

    u = (gam(i,j)          - gamma_list(j_ndex))/ ...
        (gamma_list(j_ndexp1) - gamma_list(j_ndex));

    lig(i,j) =(1.0-u)*( e_t*i_tab(i_ndex  , j_ndex ) ...
                      +t *i_tab(i_ndexp1 , j_ndex ))...
          +(   u)*( e_t*i_tab(i_ndex  , j_ndexp1) ...
                +t *i_tab(i_ndexp1 , j_ndexp1));

    end; % if-else
  end   % for j
end     % for i
return

```

11 Programs for the objective function

c:\matlab\leds\ton\func.m

function f = func(x)

```

global target_MRB   iter      setup      tilt      rot ...
                tiltperk  N_sources  prismperk  rotperk
global want_r_eta   want_t_eta want_o_uni  want_l_uni  want_SR  want_TI ...
                use_var      groupsize  grouplist  facperk   SS  inputflux

C_list = [0:5:355]';
gam_list = [0:3:90];
mag2 = (2.^(0:8));
N_veranders = size(x,2);
range = [1:N_sources];
perkb = [prismperk 180 facperk];
perko = [ 0 0 -180 0 ];
list_begin = 0;
list_end = 0;
for j=1:3
    if use_var(j) == 1
        list_begin = list_end + 1;
        list_end = list_end + groupsize(j);
        tydel = foldback(x(list_begin:list_end)', perko(j),perkb(j));
        for i=1:N_sources
            setup(i,j) = tydel(grouplist(i,j));
        end;
    end; % if
end; % for
if use_var(4) == 1
    list_begin = list_end + 1;
    list_end = list_begin;
    tilt = foldback( x(list_begin), 0, tiltperk);
end; %if
if use_var(5) == 1
    list_begin = list_end + 1;
    rot = foldback(x(list_begin), 0, rotperk);
end; %if

```

```

[HaRB, HaRB_influx, HaRB_outflux] = make_hrb(setup);

if use_var(6) == 1
    HaRB = HaRB*9.48e3/HaRB_outflux;
    [SS, SS_outflux] = total_rb(tilt, rot, HaRB);
    nuver = (SS - target_MRB)/10;
    nuver = (nuver.*nuver);
    f = sum(sum(nuver).*sin((gam_list+1.5)*pi/180));
    if (rem(iter,3)==0) | (iter==1)
        disp([ num2str(sum(use_var.*mag2)) ' ' num2str(iter) ...
            ' Meas-fit = ' sprintf('%7.4f',f/1e6) ] )
        rus=100;
        if (abs(rus*round(iter/rus)-iter)<0.00001)|(iter==2)
            rond(SS-target_MRB);
            title(['Measure of fit = ' num2str(f)])
            set(gcf,'Position',[510 50 500 310])
            drawnow
        end;
    end;
else
    [SS, SS_outflux] = total_rb(tilt, rot, HaRB);
    if inputflux == 0
        fl = SS_outflux;
    else
        fl = inputflux;
    end
    if (rem(iter,40)==0) | (iter==2)
        if rem(iter,7200) == 0
            resultaat = roadmeas(SS,fl,7)
        else
            resultaat = roadmeas(SS,fl,5);
        end
    else
        resultaat = roadmeas(SS,fl,0);
    end
    minste = [0 want_r_eta want_t_eta want_o_uni want_l_uni want_SR want_TI ];
    minste = minste - resultaat;
    minste(7) = -minste(7)/100;
    minste = (minste>0).*minste*10;

    doelwit = [0 0.96 0.81 0.42 0.72 0.45 0];
    doelwit = doelwit - resultaat;
    doelwit(7) = -doelwit(7)/100;
    doelwit(1) = -doelwit(1)/10;
    doelwit = (doelwit>0).*doelwit;
    f = doelwit*doelwit';
    f = f + minste*minste';
    disp([ num2str(sum(use_var.*mag2)) ' ' num2str(iter) ...
        ' Meas-fit = ' sprintf('%7.4f',f) ' ' sprintf('%5.2f',resultaat) ] )
    if rem(iter,100)==0
figure(2);
        rond(SS);
        view(10,40);
    end;
end
end

```

```

c:\matlab\leds\ton\foldback.m
-----
function temp = foldback(z,perko,perkb)
    z = z - perko;
    perk = perkb-perko ;
    perk2 = 2*perk ;
    temp = rem(z,perk2);

    while any(temp<0)|any(temp>perk)
        ik = find(temp < 0);
        ig = find(temp > perk);
        temp(ik) = - temp(ik);
        temp(ig) = perk2 - temp(ig);
    end;

    temp = temp +perko;

```

12 Programs for the optimizing

```

c:\matlab\leds\ton\optimis.m
-----
global iter setup stage tilt rot N_sources groupsize ...
    use_var grouplist facperk

maksiter =N_sources*200;

if size(setup,1)~=N_sources
while (size(setup,1)<N_sources)
    setup = [setup;setup]
end;
if (size(setup,1) > N_sources)
    setup = setup(1:N_sources,:);
end;
end;

fid = fopen('bestdat.m','wt');
fprintf( fid, ' tilt = %6.2f\n rot = %6.2f \n setup = [... \n',tilt,rot);
for j=1:size(setup,1)
    fprintf( fid, ' %6.2f ', setup(j,:));
    fprintf( fid, '\n');
end;
fprintf( fid, ' ]; \n');
fprintf( fid, ' %c ', setstr(37));
fprintf( fid, ' %6.0f', clock);
fprintf( fid, '%c',fullname);
fprintf( fid, '\n');
fclose(fid);

groupsize = [ N_sources N_sources N_sources];
grouplist = [1:N_sources ; 1:N_sources ; 1:N_sources]';

```

```

use_var = zeros(1,9);

optidata;

for stage = strategy
key = stage;
  setup(:,3) = round(setup(:,3));
  for i=1:8
    use_var(i) = rem(key,2);
    key = fix(key/2);
  end;
  use_var(9) = 0;
list_begin = 0;
  clear aanv
for j=1:3
  if use_var(j+6) == 1
    groupsize(j) = wantgroupsize(j);
    [grouplistie setup] = grouper(j,setup,groupsize(j));
    grouplist(:,j) = grouplistie;
    grouplist
  end
  if use_var(j) == 1
    disp([' Optimizing over the column # ' num2str(j) ' of setup']);
    list_begin = list_begin+1;
    aanv(list_begin) = setup( 1, j );
    reedsgebruik = grouplist(1, j );
    for i=2:N_sources
      if all( grouplist(i,j) - reedsgebruik ) == 1
        list_begin = list_begin+1;
        aanv(list_begin) = setup( i , j );
        reedsgebruik = [ reedsgebruik grouplist(i,j)];
      end;
    end;
  end;
end;
end;
  if use_var(4) == 1
    disp([' Optimizing over tilt ']);
    if (exist('aanv')==0)
      aanv = tilt;
    else
      aanv = [aanv tilt];
    end;
  end;
  if use_var(5) == 1
    disp([' Optimizing over rot ']);
    aanv = [aanv rot];
  end;
  if use_var(6) == 1
    disp([' Using TARGET_MRB as objective ']);
  end;
  mrye = size(aanv,2) +1;
  disp(['Number of vertices in startup simplex =' num2str(mrye) ])

  EE = eye(mrye-1);
  y = zeros(1,mrye);
  p = zeros(mrye,mrye-1);
  dx = zeros(1,mrye-1);

```

```

    dydx=zeros(1,mrye-1);
    p(1,:) = aanv;
    sprei=10*rand(mrye-1)+1;
    for i = 2:mrye
        p(i,:) = aanv +(round(rand)*2-1)*sprei(i)*EEM(i-1,:);
        dx(i-1) = p(i,i-1) - aanv(i-1);
    end;
    iter=i;
    tic
    for i = 1:size(p,1);
        y(i) = func(p(i,:));
        if i>1
            dydx(i-1) = (y(i) - y(1))/dx(i-1);
        end
    end;
    disp(' Trying a steepest descent step');
    tt = y(1)/(dydx*dydx);
    xtry = p(1,:) - tt*dydx;
    ytry = func(xtry);
    xtry2 = p(1,:) - tt*dydx/2;
    ytry2 = func(xtry2);
    if (ytry2<ytry)
        xtry=xtry2;
        ytry=ytry2;
    end
    if ytry < max(y)
        disp('Steepest descent gave a better point')
        [ii, jj] = find(y==max(max(y)));
        disp(['Replacing starting point number ' int2str(jj(1))]);
        y(jj(1)) = ytry;
        p(jj(1),:) = xtry;
    else
        disp('Steepest descent gave nothing usefull');
    end;

disp(toc)
disp(' Starting Method of Helder & Mead ');

iter = 0;
ftol = 0.001;
tic
[p,y,iter] = amoeba(p,y,ftol,maksiter);
disp(toc)
fid = fopen('bestdat.m','at');
fprintf( fid, ' tilt = %6.2f\n rot = %6.2f \n setup = [... \n',tilt,rot);
for j=1:size(setup,1)
    fprintf( fid, ' %6.2f ', setup(j,:));
    fprintf( fid, '\n');
end;
fprintf( fid, ' ]; \n');
fprintf( fid, ' %c ', setstr(37));
fprintf( fid, ' %6.0f', clock);
fprintf( fid, ' After stage %3d',stage);
fprintf( fid, ' Ave of best: %12.4f\n', sum(y)/length(y) );
fclose(fid);
end

```

```

c:\matlab\leds\ton\grouper.m
-----
function [gri, tabel] = grouper(jj,setup,mm)

epsi = 1e-10;
N_sources = size(setup,1);
m = size(setup,2);
    % Make a copy of SETUP to use as working space, with the
    % (m+1)st column a list of indices which will refer to the group.
houtabel = [setup [1:N_sources]'];

while max(houtabel(:,m+1)) > mm
    % From a new tabel of only the first element in each group
    tabel = [houtabel(1,jj) 1];          % : [first element 1]
    groupindices = houtabel(1,m+1);      % : index of group of first
    for i=2:N_sources
        if all(groupindices ~= houtabel(i,m+1)) % if the element oes not have same index
            tabel = [tabel
                    houtabel(i,jj) i];      % [its_value its_real_index]
            groupindices = [groupindices houtabel(i,m+1)];
        end
    end
    % How many groups are left? ---> n
    n = size(tabel,1);
    % How far apart are the different items?
    clear distances
    for i = 1:n
        for j = 1:n
            if i < j
                distances(i,j) = abs(tabel(i,1)-tabel(j,1));
            else
                distances(i,j) = 1e9;          % Ridiculous values on the diagonal
            end
        end
    end
    % What is the smallest distance between values?
    kleinste = min(min(distances));
    % Where does it occur?
    [ikl jkl] = find(abs(distances - kleinste)<epsi);
    % What is the index of the first occurence --- to be used as refence group
    i1 = ikl(1);
    % What is the index of the first one among those that were the closest?
    i2 = jkl(1);
    % This one (and only this one ) must be erased from TABEL and
    % assigned to the same group index of the one to which
    % it is coupled inserted in HOUTABEL
    iw1 = tabel(i1,2);
    iw2 = tabel(i2,2);
    groepnommer1 = min( houtabel(iw1,m+1) , houtabel(iw2,m+1) );
    groepnommer2 = max( houtabel(iw1,m+1) , houtabel(iw2,m+1) );
    antw = [groepnommer1 groepnommer2];
    tel1 = 0;
    som1 = 0;
    tel2 = 0;
    for i=1:N_sources
        if abs(houtabel(i,m+1) - groepnommer1) < epsi

```

```

        tell = tell+1;
        som1 = som1 +houtabel(i,jj);
    elseif abs(houtabel(i,m+1) - groepnommer2) < epsi
        tel2 = tel2 +1;
        if tel2 == 1          % Take only one from the other group!
            som2 = houtabel(i,jj);
            houtabel(i,m+1) = groepnommer1;
        end
    else
        end;
    end;
end
waarde = (som1+som2)/(tell+1);
antw = [antw waarde];
for i = 1:N_sources
    if abs(houtabel(i,m+1) - groepnommer1)< epsi
        houtabel(i,jj) = waarde;
    elseif (houtabel(i,m+1) >= round(groepnommer2 ) ) & (tel2 == 1)
        houtabel(i,m+1) = houtabel(i,m+1) - 1;
    end;
end;
end;
end;

tabel = houtabel(:,1:(size(setup,2)));
gri = houtabel(:,m+1);

```

13 Programs for the road measurements

c:\matlab\leds\ton\roadmeas.m

```

-----
function sukses = roadmeas(tans,tans_influx, key,tonnaam);
% ROADMEAS input --- tans:          an IES-table.
%                   --- tansinflux: the inputflux in the MRB
% input:  key = 1 : full data to screen
%         key = 2 : full data to filename rN_D.T.txt, with M,D,T one or two digit
%                   numbers containing the month, the day and the hour of file creation
%         key = 4 : draw picture of road illumination
%         key = 8 : draw contour maps of road illumination
%         key = 16: Write all luminances to file
%         Add values of key, if more than one action is wanted.
% FORM:   function sukses = roadmeas(tans,tans_influx, key);

global newroad    tangam_list  beta_list  rtab
global beta_FPBi r_valuesBid  beta_FPBi2 r_valuesB2d ...
      beta_FPBi  r_valuesBid  Poles      ...
      C_FP       cosg_FP      tang_FP    g_FP      d2FP      ...
      K1         LW1         LW2         K2         N_grid    ...
      N_K1       N_LW1       N_LW2       N_K2       N_obs     ...
global K1ind     LW1ind     LW2ind     K2ind     Roadind    ...
      N_xP       N_yP       xP         yP         x_displ_obs ...
      F_height  GS         S_poles  W_road    W_total    ...
      gam_list  W_road     indices   ...
      C1i       gam1i     veil_koef1 ...
      C2i       gam2i     veil_koef2 ...

```

```

        want_lum tilt      fullname rot      setup
C_list = [0:5:355];
gam_list = [0:3:90];

%disp(['newroad' int2str(newroad)]);
if (newroad == 1) %(exist('xP') == 0)
    disp(' I do not yet have all the info .... COMPUTING ..... ');
    roadinit
    newroad = 0;
end;

tempC = [C_list 360];
temptans = [tans
            tans(1,:)];
tempg = [gam_list 180];
temptans = [temptans zeros(size(temptans(:,1)))];
I_FP = interp2( tempC, tempg', temptans', C_FP, g_FP);
E_FP = I_FP./d2FP.*cosg_FP;

% Total horizontal illumination at all gridpoints
somh = zeros(N_xP,N_yP);

for i = 1:size(Poles,1)
    somh = somh + E_FP(indices(i,:),:);
end;

% MEASURES THAT HAS TO DO WITH ILLUMINATION Paragraph 5.2.1

EhK1 = mean(mean( somh(:, K1ind ) ));
EhLW1 = mean(mean( somh(:, LW1ind ) ));
EhLW2 = mean(mean( somh(:, LW2ind ) ));
EhK2 = mean(mean( somh(:, K2ind ) ));

SR1 = (EhK1)/(EhLW1);
SR2 = (EhK2)/(EhLW2);

Phi = fluxies(C_list,gam_list,tans);
eta_road = ( EhLW1*LW1+EhLW2*LW2 )*(S_poles)/Phi ;
eta_total = (EhK1*K1+EhLW1*LW1+EhLW2*LW2+EhK2*K2)*(S_poles)/Phi ;

% MEASURES THAT HAS TO DO WITH LUMINANCE (OVERALL!!!) Paragraph 5.2.2 - part1
L_FB1 = r_valuesB1d .* I_FP ;
L_FB2 = r_valuesB2d .* I_FP ;
somLB1 = zeros(N_xP,N_yP);
somLB2 = zeros(N_xP,N_yP);
for i = 1:size(Poles,1)
    somLB1 = somLB1 + L_FB1(indices(i,:),:);
    somLB2 = somLB2 + L_FB2(indices(i,:),:);
end;

L_ave1 = mean(mean(somLB1(:,Roadind)));
L_ave2 = mean(mean(somLB2(:,Roadind)));
L_min1=min(min(somLB1(:,Roadind)));
L_min2=min(min(somLB2(:,Roadind)));
U_0_1 = L_min1/L_ave1;
U_0_2 = L_min2/L_ave2;

```

```

neededflux = want_lum/(min([L_ave1,L_ave2]))*Phi;

% MEASURES THAT HAS TO DO WITH lengthwise uniformity in LUMINANCE Paragraph 5.2.2 - part2

L_FBi = r_valuesBid .* I_FP(:,Roadind) ;
somLBi = zeros( N_xP, (N_LW1+N_LW2) );
for i = 1:size(Poles,1)
    somLBi = somLBi + L_FBi(indices(i,:),:);
end;

L_avei = max(somLBi) ;
L_mini = min( somLBi);
U_length_i = L_mini./L_avei;

convertfac = neededflux/Phi;
tans = tans*convertfac;
liggie = illoutve( C_list, gam_list, tans, C1i, gam1i );
L_v1 = veil_koef1.*liggie;
liggie = illoutve( C_list, gam_list, tans, C2i, gam2i );
L_v2 = veil_koef2.*liggie;
TI1 = 65*sum(L_v1)/(L_ave1*convertfac)^0.8 ;
TI2 = 65*sum(L_v2)/(L_ave2*convertfac)^0.8 ;

Input_lum_need = neededflux*tans_influx/Phi;
kL_cdm2 = Input_lum_need/1000/want_lum;
sukses = [kL_cdm2 eta_road eta_total min(U_0_1,U_0_2) ...
          min(U_length_i) min(SR1,SR2) max(TI1,TI2)];

cf = convertfac;

if (rem(key,2)==1) % eerste deling : screen print
    if nargin==4
        writeall(0, tilt, rot, setup, neededflux, ...
                 Input_lum_need, eta_total, kL_cdm2, eta_road, SR1, ...
                 SR2, L_ave1*cf, L_ave2*cf, U_0_1, U_0_2, ...
                 U_length_i, TI1, TI2, EhK1*cf, EhLW1*cf, ...
                 EhLW2*cf, EhK2*cf, tonnaam );
    else
        writeall(0, tilt, rot, setup, neededflux, ...
                 Input_lum_need, eta_total, kL_cdm2, eta_road, SR1, ...
                 SR2, L_ave1*cf, L_ave2*cf, U_0_1, U_0_2, ...
                 U_length_i, TI1, TI2, EhK1*cf, EhLW1*cf, ...
                 EhLW2*cf, EhK2*cf );
    end;
end

key = fix(key/2); % Tweede deling : file print
if (rem(key,2)==1)
    if nargin==4
        writeall(1, tilt, rot, setup, neededflux, ...
                 Input_lum_need, eta_total, kL_cdm2, eta_road, SR1, ...
                 SR2, L_ave1*cf, L_ave2*cf, U_0_1, U_0_2, ...
                 U_length_i, TI1, TI2, EhK1*cf, EhLW1*cf, ...
                 EhLW2*cf, EhK2*cf, tonnaam );
    else
        writeall(1, tilt, rot, setup, neededflux, ...

```

```

        Input_lum_need, eta_total,    kL_cdm2,    eta_road,    SR1,    ...
        SR2,        L_ave1*cf,    L_ave2*cf,    U_0_1,    U_0_2,    ...
        U_length_i,    TI1,        TI2 ,        EhK1*cf,    EhLW1*cf,    ...
        EhLW2*cf,    EhK2*cf ) ;

end;
end

key = fix(key/2); % derde deling : draw road
if (rem(key,2)==1)
    drawroad(1,S_poles, xP, yP, somh, somLB1, somLB2, K1, LW1, LW2, K2);
end

key = fix(key/2); % vierde deling : draw contour
if (rem(key,2)==1)
    drawroad(0,S_poles, xP, yP, somh, somLB1, somLB2, K1, LW1, LW2, K2);
end

key = fix(key/2); % vyfde deling : write lums
if (rem(key,2)==1)
    writroad( convertfac*somLB1,convertfac*somLB2,convertfac*somLBi, ...
        somh*convertfac, K1,LW1,LW2,K2,K1ind,LW1ind,LW2ind,K2ind,Roadind);
end

c:\matlab\leds\ton\roadinit.m
-----
W_total = K1+LW1+LW2+K2
W_road  = LW1+LW2
nn=size(Poles,1)
S_poles = mean(Poles(2:nn,1)-Poles(1:(nn-1),1))

N_grid    = 10 ;                % Number of grid points on road, same grid
                                % spacing will be used on kerbs)

x_coord_obs = x_displ_obs +S_poles/2
                                % x_coord_obsTI must be such that the angle down
                                % from the luminaire must be 20 deg.

ell = (F_height-1.5)/tan(20*pi/180);
x_coord_obsTI = ell +2*S_poles

disp1 = x_coord_obs ;
disp2 = x_coord_obs - S_poles;
hoek1 = atan2(1.5, disp1)*180/pi;
hoek2 = atan2(1.5, disp2)*180/pi;
disp(['Observed field lies between ' num2str(hoek1) ' deg and ' num2str(hoek2) ' deg']);

                                % Grid points
GS = W_road/N_grid;
N_K1 = round(K1/GS);
N_LW1 = round(LW1/GS);
N_LW2 = round(LW2/GS);
N_K2 = round(K2/GS);
while (W_road/(N_LW1+N_LW2) ~= GS)
    N_grid = N_LW1+N_LW2;
    GS = W_road/N_grid;
    N_K1 = round(K1/GS);

```

```

    N_LW1 = round(LW1/GS);
    N_LW2 = round(LW2/GS);
    N_K2 = round(K2/GS);
    disp('WARNING!! REDEFINING FOR REASONS OF SYMMETRY!')
    disp([' N_grid = ' int2str(N_grid) ' Grid Spacing = ' num2str(GS)] );
end

N_obs = N_LW1 + N_LW2;
N_yP = N_K1 + N_obs + N_K2;
N_xP = round( (S_poles)/GS );

K1ind = 1:N_K1;
LW1ind = N_K1 + (1:N_LW1);
LW2ind = N_K1 + N_LW1 + (1:N_LW2);
K2ind = N_K1 + N_LW1 + N_LW2 + (1:N_K2);
Roadind = [LW1ind LW2ind];

y_K1 = GS/2:GS:K1;
y_res = GS/2:GS:(W_road + K2);
yP = [-fliplr(y_K1) y_res];
xP = linspace( GS/2, S_poles-GS/2, N_xP)';

B1 = [x_coord_obs LW1/2 1.5];
B2 = [x_coord_obs LW1+LW2/2 1.5];

BB1 = [x_coord_obsTI LW1/2 1.5];
BB2 = [x_coord_obsTI LW1+LW2/2 1.5];

% Position vectors and quantities relative to luminaires
% VECTORS FROM EACH luminaire TO EACH grid point
% Number the F's from 1 (at -4*S_poles from origin) to 7 (at 2*S_poles from origin)
% Store the information for each gridpoint in a submatrix of the same size as the
% gridpoints, for luminaire F_1 in rows 1:N_xP, for F_2 in (N_xP+1):(2*N_xP), etc.
% for the above use " name( indices(i,:),:)"
% where

index = [1:N_xP];
indices = index;
for i=1:(size(Poles,1)-1)
    indices = [ indices
               index + i*N_xP];
end

% xFP : x-coord
% yFP : y-coord
% zFP : z-coord : all the same, = -F_height
% d2FP : squares of following distances:
% dFP : distances F to P
% d2OP : squared of following distances :
% dOP : distances point under F to P
% C_FP : C angles from luminaire to point
% g_FP : gamma angles from luminaire to point
% as needed we will compute

% THE POSITION VECTORS _FP_ AND RELATED QUANTITIES
for i=1:size(Poles,1)
    xFP(indices(i,:),1:N_yP) = (xP-Poles(i,1))*ones(size(yP));

```

```

        yFP(indices(i,:),1:N_yP) = ones(size(xP))*(yP-Poles(i,2));
    end;
    d20P = xFP.^2 + yFP.^2;
    d2FP = d20P + F_height.^2;
    d0P = sqrt(d20P);
    dFP = sqrt(d2FP);

    C_FP = atan2( yFP , xFP )/pi*180;
    C_FP = C_FP +360.*(C_FP<0);
    for i=1:size(Poles,1)
        if ( Poles(i,2) > LW1 )
            C_FP(indices(i,:),1:N_yP) =C_FP(indices(i,:),1:N_yP)-180;
        end;
    end;
    C_FP = C_FP +360.*(C_FP<0);

    cosg_FP = (F_height)./dFP;
    g_FP = acos(cosg_FP) ;
    tang_FP = tan(g_FP);
    g_FP = g_FP/pi*180;

    % POSITION VECTORS AND quantities relative to Observers 1 and 2
    % xB1P, xB2P : x-coord P rel to Observer_i
    % yB1P, yB2P : y-coord P rel to Observer_i
    %          : z-coord P rel to Observer_i --- all the same
    % dBriP      : distance of point on road under Bi to P
    % dBiP       : distance Bi to P

    xB1P = ( xP-B1(1) )*ones( size(yP) ) ;
    yB1P = ones(size(xP))*( yP - B1(2) ) ;
    dB1rP = sqrt( xB1P.^2 +yB1P.^2);
    for i=1:size(Poles,1)
        beta_FPBi(indices(i,:),1:N_yP) = acos( ...
            -( xFP(indices(i,:),:) .* xB1P +...
                yFP(indices(i,:),:) .* yB1P ) ./...
            ( d0P(indices(i,:),:) .* dB1rP ) )/pi*180;
    end

    xB2P = ( xP-B2(1) )*ones( size(yP) ) ;
    yB2P = ones(size(xP))*( yP - B2(2) ) ;
    dB2rP = sqrt( xB2P.^2 +yB2P.^2);
    for i=1:size(Poles,1)
        beta_FPBi(indices(i,:),1:N_yP) = acos( ...
            ( -xFP(indices(i,:),:) .* xB2P +...
                yFP(indices(i,:),:) .* yB2P ) ./...
            ( d0P(indices(i,:),:) .* dB2rP ) )/pi*180;
    end

    xBiP = ( xP-B1(1) )*ones( size(yP([LW1ind LW2ind])) ) ) ;
    % everyone yBiP is zero: lengthwise
    dBirP = abs( xBiP ); % +yB1P.^2);
    for i=1:size(Poles,1)
        beta_FPBi(indices(i,:),1:(N_LW1 +N_LW2) ) = acos( ...
            ( -xFP(indices(i,:),Roadind) .* xBiP ) ./...
            ( d0P(indices(i,:),Roadind) .* dBirP ) )/pi*180;
    end
end

```

```

maks=max(tangam_list);
tang_FP = tang_FP.*( tang_FP <= maks) +maks.*( tang_FP > maks);
r_valuesB1 = interp2( beta_list , tangam_list' , rtab' , beta_FP1 , tang_FP)/10000;
r_valuesB2 = interp2( beta_list , tangam_list' , rtab' , beta_FP2 , tang_FP)/10000;
tang_FP = tang_FP(:,Roadind);
r_valuesBi = interp2( beta_list , tangam_list' , rtab' , beta_FPBi , tang_FP)/10000;

r_valuesBid = r_valuesBi ./ (F_height.^2);
r_valuesB1d = r_valuesB1 ./ (F_height.^2);
r_valuesB2d = r_valuesB2 ./ (F_height.^2);

B1P = [-1.5/tan(1*pi/180)    0   -1.5];
B2P = [-1.5/tan(1*pi/180)    0   -1.5];
dd1 = sqrt(B1P*B1P');
dd2 = sqrt(B2P*B2P');
for i = 1:size(Poles,1)
    FiB1 = BB1 - [Poles(i,1) Poles(i,2) F_height];
    FiB2 = BB2 - [Poles(i,1) Poles(i,2) F_height];
    dFiB1 = sqrt(FiB1*FiB1');
    dFiB2 = sqrt(FiB2*FiB2');
    costheta1(i) = -FiB1*B1P'/(dFiB1 * dd1);
    theta1(i) = acos(costheta1(i))/pi*180;
    costheta2(i) = -FiB2*B2P'/(dFiB2 * dd2);
    theta2(i) = acos(costheta2(i))/pi*180;
    gam1i(i) = 180/pi*acos(-FiB1(3)/dFiB1);
    gam2i(i) = 180/pi*acos(-FiB2(3)/dFiB2);
    C1i(i) = atan2(FiB1(2),FiB1(1))*180/pi;
    C2i(i) = atan2(FiB2(2),FiB2(1))*180/pi;
    if (Poles(i,2) > LW1 )
        C1i(i) = C1i(i) - 180;
        C2i(i) = C2i(i) - 180;
    end
    veil_koef1(i) = 10*costheta1(i)/((dFiB1*theta1(i))^2);
    veil_koef2(i) = 10*costheta2(i)/((dFiB2*theta2(i))^2);
end;
C1i = C1i +360.*(C1i<0);
C2i = C2i +360.*(C2i<0);

disp('Angle theta, C, gamma, and 1000*veiling coefficient for first observer for II');
[theta1' C1i' gam1i' 1000*veil_koef1']

disp('Angle theta, C, gamma, and 1000*veiling coefficient for second observer for II');
[theta2' C2i' gam2i' 1000*veil_koef2']

```

14 Program for IES files

```

c:\matlab\leds\ton\vormies.m
-----
function sukses = vormies(ies,stringie)
% VORMIES writes the IES-info to a file named NAAM,
% in the same form as the original IES-files

```

```

global C_list gam_list
flux = fluxies(C_list, gam_list, ies);

fid = fopen(stringie,'wt');
fprintf( fid, 'IES\A91\n');
fprintf( fid, '@IES LMT LICHTMESSTECHNIK GMBH BERLIN Version 02/92\n');
fprintf( fid, 'LVE01321.LMT ***FLUX ');
fprintf( fid, '%6.4f ',flux/1000);
fprintf( fid, ' LM***\n');
fprintf( fid, 'LVE01321.LMT\n');
fprintf( fid, 'MRB 19mm DEFL:vari SPREAD:combined VBU\n');
fprintf( fid, 'Numerous sources, combined optics\n');
fprintf( fid, 'LEDs ambe 292 D1-2*8 ALU-inlay +LENS\n');
fprintf( fid, 'file December 97 (J Spoelstra)\n');
fprintf( fid, '\n');
fprintf( fid, 'LumiLeds\n');
fprintf( fid, '%c',date);
fprintf( fid, '\n');
fprintf( fid, 'TILT=comb\n');
fprintf( fid, '1 1000 1 91 360 1 2 0 0 0 \n');
fprintf( fid, '1 1 0\n');
for i=0:90
    fprintf( fid, ' %d',i);
end;
fprintf( fid, ' \n');
for i=0:119
    fprintf( fid, ' %d',i);
end;
fprintf( fid, ' \n');
for i=120:239
    fprintf( fid, ' %d',i);
end;
fprintf( fid, ' \n');
for i=240:359
    fprintf( fid, ' %d',i);
end;
fprintf( fid, '\n');
j=[0.001 1:89 89.995]
ies = [ies
        ies(1,:)];
C_temp = [C_list
          360];
for i = [0.001 1:359]
    disp(num2str(i))
    lig = interp2(C_temp,gam_list',ies', i*ones(size(j)), j);
    fprintf( fid, ' %d',round(lig));
    fprintf( fid, '\n');
end;
sukses = fclose(fid);

```

15 Utilities

c:\matlab\leds\tools\amoeba.m

```

-----
function [p,y,iter] = amoeba(p,y,ftol,maksiter)
global iter
alfa = 1.0;
beta = 0.5;
gamma = 2.0;
ndim = size(p,2);
itmaks= maksiter; %500 ; %3*ndim; %500;
mpts = ndim +1 % aantal punte
iter = 0;

% BepaalHoog2deHoogLaag;
[iHoog,i2deHoog,iLaag] = bh2hl(mpts,y);
rtolteller=1000000;
rtolnoemer=1000000;
while iter<=itmaks % (rtolteller > ftol*rtolnoemer)&(rtolnoemer>1e-10) ;
    iter = iter +1;
    % disp([' Iter = ' num2str(iter)])
    % RefleksieDeurMidptVanVlakTeenoerHoog;
    % disp('rdmvvth')
    ibeh_hoog = [1:(iHoog-1) (iHoog+1):mpts];
    pbar = sum(p(ibeh_hoog,:))/ndim ;
    pr = (1.0 +alfa)*pbar - alfa*p(iHoog,:);
    ypr = func(pr);
    if ypr <= y(iLaag) % RefleksieMetEkspansie
        % disp(' 1')
        prr = gamma*pr +(1.0-gamma)*pbar;
        yprr = func(prr);
        if yprr < y(iLaag) %1
            % disp(' +2')
            p(iHoog,:) = prr;
            y(iHoog) = yprr;
        else %1
            p(iHoog,:) = pr;
            y(iHoog) = ypr;
        end %1
    elseif ypr >= y(i2deHoog) % Strategie3En4
        if ypr < y(iHoog)
            p(iHoog,:) = pr;
            y(iHoog) = ypr;
        end;
        prr = beta*p(iHoog,:) +(1.0-beta)*pbar;
        yprr = func(prr);
        if yprr < y(iHoog) % AanvaarKontraksie;
            p(iHoog,:)=prr;
            y(iHoog) = yprr;
        else % TrekSaamOmLaagstePunt;
            for i = 1:mpts
                if i ~= iLaag
                    pr = (p(i,:)+p(iLaag,:))/2;
                    p(i,:) = pr;
                    y(i) = func(pr);
                end;
            end;
        end % Strat34
    else
        % disp(' M')

```

```

        p(iHoog,:) = pr;
        y(iHoog) = ypr;
    end;
        % BepaalHoog2deHoogLaag;
    [iHoog,i2deHoog,iLaag] = bh2hl(mpts,y);
        % Ontsnaproetes;
    rtolnoemer = ( abs(y(iHoog))+abs(y(iLaag)) )/2;
    rtolteller = abs(y(iHoog) -y(iLaag));
    if (rtolteller < ftol*rtolnoemer)|(rtolnoemer<1e-10)
        iter = itmaks
        disp(' Stop --- normaal')
    end
    if iter >= itmaks
        disp(' Stop in AMOEBA - te veel iteraties, behalve as stop normaal');
%       disp(' Finale antwoord:');
%       p
%       y
        yres=func(p(iLaag,:))
        return
    end
end;

```

c:\matlab\leds\tools\bh2hl.m

```

-----
function [iHoog,i2deHoog,iLaag] = bh2hl(mpts,y);
% BepaalHoog2deHoogLaag;
    iLaag = 1;
    if y(1) > y(2)
        iHoog = 1;
        i2deHoog = 2;
    else
        iHoog = 2;
        i2deHoog = 1;
    end;
    for i = 1:mpts
        if y(i) < y(iLaag)
            iLaag = i;
        end
        if y(i) > y(iHoog)
            i2deHoog = iHoog;
            iHoog = i;
        elseif ( y(i) > y(i2deHoog) ) & ( i ~= iHoog )
            i2deHoog = i;
        end;
    end;
end;

```

c:\matlab\leds\tools\drawroad.m

```

-----
function sukses = drawroad(sleutel,S_poles,xP,yP,somh,somLB1,somLB2,K1,LW1,LW2,K2)

somx = somh;
txP = xP;
la = round(max(xP)+0.1);
[X,Y] = meshgrid(txP,yP);

som1 = somLB1;

```

```

som2 = somLB2;

figure(1)
subplot(3,1,1)
if sleutel == 1
    surf( X,Y,somx' )
else
    contour(X,Y,somx',10)
end
line([0 0],[0,0]);
text(0,0,10,'*');
line([1a 1a],[0,0],[0,10]);
text(1a,0,10,'*');
line([-0 1a+1],[ 0 0], [2 2 ]);
line([-0 1a+1],[LW1+LW2 LW1+LW2 ], [2 2 ])
line([-0 1a+1],[ -K1 -K1], [0.3 0.3 ]);
line([-0 1a+1],[LW1+LW2+K2 LW1+LW2+K2], [0.3 0.3 ])
view(2)
shading interp;
colormap(hot)
title('Horizontal illumination, E_h')

subplot(3,1,2)
if sleutel == 1
    surf( X,Y,som1')
else
    contour(X,Y,som1',10)
end
line([0 0],[0,0],[0,10]);
text(0,0,10,'*');
line([1a 1a],[0,0],[0,10]);
text(1a,0,10,'*');
line([-0 1a+1],[ 0 0], [2 2 ]);
line([-0 1a+1],[LW1+LW2 LW1+LW2], [2 2 ])
line([-0 1a+1],[ -K1 -K1], [0.3 0.3 ]);
line([-0 1a+1],[LW1+LW2+K2 LW1+LW2+K2], [0.3 0.3 ])
view(2)
shading interp;
colormap(hot)
title('Luminance in direction of first observer')

subplot(3,1,3)
if sleutel == 1
    surf(X,Y,som2')
else
    contour(X,Y,som2',10)
end
% surf(X,Y,som2')
line([0 0],[0,0],[0,10]);
text(0,0,10,'*');
line([1a 1a],[0,0],[0,10]);
text(1a,0,10,'*');
line([-0 1a+1],[ 0 0], [2 2 ]);
line([-0 1a+1],[LW1+LW2 LW1+LW2 ], [2 2 ])
line([-0 1a+1],[ -K1 -K1], [0.3 0.3 ]);
line([-0 1a+1],[LW1+LW2+K2 LW1+LW2+K2], [0.3 0.3 ])
view(2)

```

```

shading interp;
colormap(hot)
title('Luminance in direction of second observer')

set(gcf,'Position',[20 130 960 600])

drawnow

end

c:\matlab\leds\tools\rond.m
-----
function ha = rond(i3)
%ROND
n = 72;
m = 31;
C = (0:n)/n*2*pi;
gamma = (0:3:90)';
sinC = sin(C);
cosC = cos(C);
x = gamma * cosC;
y = gamma * sinC;
z = i3';
z(:,73)=z(:,1);
surf(x,y,z)
shading interp

text(100,0,100,'C=0')
text(-15,95,100,'C=90')
view(2)
%colorbar
xx=[-95:95];
xx8=0.84*xx;
xx5=0.50*xx;
zz=100*ones(size(xx));
line( xx,0*xx ,zz);
line(0*xx, xx ,zz);
line( xx8, xx5,zz);
line( xx5, xx8,zz);
line( -xx8, xx5,zz);
line( -xx5, xx8,zz);
ccc='.';
for i=-pi:0.05:pi
    xi=cos(i);
    yi=sin(i);
    hh=['text(' num2str(80*xi) ',' num2str(80*yi) ', 100,' 39 ccc 39 ')']];
    eval(hh );
    hh=['text(' num2str(60*xi) ',' num2str(60*yi) ', 100,' 39 ccc 39 ')']];
    eval(hh );
    hh=['text(' num2str(40*xi) ',' num2str(40*yi) ', 100,' 39 ccc 39 ')']];
    eval(hh );
    hh=['text(' num2str(20*xi) ',' num2str(20*yi) ', 100,' 39 ccc 39 ')']];
    eval(hh );
end
ha = 1;

```

```

return

c:\matlab\leds\tools\writeall.m
-----
function sukses = writeall(...
    sleutel, tilt, rot, setup, neededflux, Input_lum_need, ...
    eta_total, kL_cdm2, eta_road, SR1, SR2, L_ave1, ...
    L_ave2, U_0_1, U_0_2, U_length_i, TI1, TI2, ...
    EhK1, EhLW1, EhLW2, EhK2, tonnaam)

global SORS_FLX fullname K1 LW1 LW2 K2 ...
    want_t_eta want_r_eta want_o_uni want_l_uni want_TI want_SR...
    Poles F_height fullname1 fullname2 fullname3 fullname4 fullname5
if sleutel==0
    fid = 1;
else
    klok = clock;
    stringie = ['c:\matlab\results\r' num2str(klok(2:4)) '.txt'];
    fid = fopen(stringie,'wt');
end

fprintf(fid,'\n\n ----- Luminaire data ----- ');
fprintf(fid,'%c',date);
fprintf(fid,'\n ');
if nargin==23
    fprintf(fid,'\n ');
    fprintf(fid,'%c',tonnaam);
    fprintf(fid,'\n');
else
    fprintf(fid,' Optical type (name of datafile): ');
    fprintf(fid,' Optical type (name of datafile): \n ');
    for i=1:size(SORS_FLX,2)
        eval(['fullname = fullname' int2str(i) '']);
        fprintf(fid,'%c',fullname);
        fprintf(fid,' corresponds with third factor = %2d',i-1);
        fprintf(fid,'\n ');
    end
    fprintf(fid,' Tilt of the luminaire: %5.1f deg\n',tilt);
    fprintf(fid,' Rotation of the luminaire: %5.1f deg\n',rot );
    fprintf(fid,' Deflection, placement and third factor:\n' );
    fprintf(fid,' %7.1f %7.1f %7.1f\n',setup' );
end;
fprintf(fid,'\n ----- Road -----\n');
fprintf(fid,...
    ' Left kerb %5.1fm Left lane %5.1fm Right lane %5.1fm Right kerb %5.1fm \n',...
    K1,LW1,LW2,K2);

fprintf(fid,'\n ----- Poles -----\n');
pale = [Poles F_height*ones(size(Poles,1),1)];
fprintf(fid,' %7.1f %7.1f %7.1f \n',pale' );

fprintf(fid,'\n ----- Efficiencies -----\n');
fprintf(fid,' Output lumens needed per pole %8.2f\n', neededflux );
fprintf(fid,' Input lumens needed per pole %8.2f\n', Input_lum_need);
fprintf(fid,' kilo-Lumens needed per cd/m^2 %8.2f\n', kL_cdm2);
fprintf(fid,' Total efficiency %8.1f %c\n', eta_total*100,'%');

```

```

fprintf(fid,' Road efficiency                %8.1f %c\n', eta_road*100 , '%' );
fprintf(fid,' Eh (average) (lux)\n');
fprintf(fid,'     Left kerb %6.1f \n', EhK1);
fprintf(fid,'     Left lane %6.1f \n', EhLW1);
fprintf(fid,'     Right lane %6.1f \n', EhLW2);
fprintf(fid,'     Right kerb %6.1f \n', EhK2);

fprintf(fid,' Left surround ratio          %8.1f %c\n', SR1*100 , '%' );
fprintf(fid,' Right surround ratio           %8.1f %c\n', SR2*100 , '%' );

fprintf(fid,' \n ----- Luminances -----\n');
fprintf(fid,' Average luminance for first observer %8.2f cd/m^2\n',L_ave1 );
fprintf(fid,' Average luminance for second observer %8.2f cd/m^2\n',L_ave2 );
fprintf(fid,' Overall uniformity for first observer %8.2f\n',U_0_1);
fprintf(fid,' Overall uniformity for second observer %8.2f\n',U_0_2);
fprintf(fid,' Line uniformities ');
fprintf(fid,' %4.2f',U_length_i);
fprintf(fid,' \n Threshold increment for first observer %8.2f %c\n',TI1, '%');
fprintf(fid,' Threshold increment for second observer %8.2f %c\n',TI2, '%');

fprintf(fid,...
'\n Wanted values: SR >= %6.2f  U_o >= %6.2f,  U_l >= %6.2f,  TI <= %6.1f %c \n', ...
want_SR*100, want_o_uni, want_l_uni, want_TI, '%');

```

c:\matlab\leds\tools\writroad.m

```

-----
function sukses = writroad( somLB1,somLB2,somLBi,...
                           somh, K1,LW1,LW2,K2,K1ind,LW1ind,LW2ind,K2ind,Roadind)

stringie = input('Name of (new) file to use for writing all luminances? ', 's')
fid = fopen(stringie,'wt')
lank = size(somLB1,1);

fprintf(fid,' Horizontal illumination on the KERBS, ...
           (must be divided by 1000 to get lux)');
fprintf(fid,' \n ----- LEFT KERB ----- Width: %4.1f m \n',K1);
for i=1:lank
    fprintf(fid,' %6.0f', 1000*somh(i,K1ind));
    fprintf(fid,' \n');
end
fprintf(fid,' \n ----- RIGHT KERB ----- Width: %4.1f m \n',K2);
for i=1:lank
    fprintf(fid,' %6.0f', 1000*somh(i,K2ind));
    fprintf(fid,' \n');
end

fprintf(fid,' \n ----- ROAD ----- Width: %4.1f m (Observer 1) \n',LW1+LW2);
fprintf(fid,' All quantities are in must be divided by 1000 to get cd/m^2\n');
for i=1:lank
    fprintf(fid,' %6.0f', 1000*somLB1(i,Roadind));
    fprintf(fid,' \n');
end

fprintf(fid,' \n ----- ROAD ----- Width: %4.1f m (Observer 2) \n',LW1+LW2);
fprintf(fid,' All quantities must be divided by 1000 to get cd/m^2\n');

```

```

for i=1:lank
    fprintf(fid, ' %6.0f', 1000*somLB2(i,Roadind));
    fprintf(fid, ' \n');
end

fprintf(fid, '\n\n ----- 10 Observers, spaced across road \n');
fprintf(fid, ' All quantities must be divided by 1000 to get cd/m2\n');
fprintf(fid, '\n ----- ROAD ----- Width: %4.1f m \n',LW1+LW2);
for i=1:lank
    fprintf(fid, ' %6.0f', 1000*somLBi(i,:));
    fprintf(fid, ' \n');
end

fclose(fid);
sukses=fid;

```