ANNEX 1

ALGORITHM AND FUNCTIONS

ANNEX 1 75

PROFILES 77

Read values from data text file 77
Compute a linear regression using polyfit 77
Horizontalisation if needed 77
Verification of correctness in *horizontalisation* 78
Calculation of Py and lc0; Print Table 1 of parameters for yvec2 78
Plot the data and label the plot 78
Iteration to find last value of lc 78
Plot data 79
Save data 80
Save plots 81
Ask for new lc 81

MAKEVEC.M 82

Get name of folder and file 82
Open file 82
Save on vectors 82

ISMEQUALTOZERO.M 83

Regression line of xa and ya 83
Return 83

CALCRY.M 84

Determination of intervals and Ry 84
Tables 84

CALCRW.M 85

Gaussian filter vectors 85
Convolution 85
Obtaining roughness profile: yvecr = yvec - yvecw 85

CUTFUNC.M 86

Calculus via iterative function of final cut-off length, lc 86
Calculus of final yvecw and yvecr 86
Print Table of parameters for yvecr 86

CALCITE.M **87**
Cut-off, evaluation length and n 87
Obtaining Roughness Profile 87
Intervals of study 87
Calculus of Ra and new lc (lc2) 88
Iteration 88
CALCINT.M **90**
lt and lv 90
Indexes 90
Plot 90
CALCRARQ.M **91**
Obtaining limits for xvecln and yvecrln 91
Main calculation of Ra and Rq 91
VALRALC.M **92**
ROUGHCLASS.M **93**

Profiles

```
% This script differentiates roughness from waviness in profiles
% according to frequency studies
```

Read values from data text file

```
[xvec, yvec1, name] = makevec();

%The x-axis is in the 1st column, horizontal values
%The y-axis is in the 2nd column, height values

[r, c] = size(xvec);
fprintf('%5.0f rows and %0.0f columns for xvec.\n', r, c);

xmin = min(xvec);
xmax = max(xvec);
xcent = (xmax-xmin)/2 + xmin;
ymin = min(yvec1);
ymax = max(yvec1);

fprintf('xmin = %5.3f, xmax = %5.3f, center = %5.3f (mm).\n', xmin, xmax, xcent);
fprintf('ymin = %5.3f and ymax = %5.3f (micrometers).\n', ymin, ymax);
```

Compute a linear regression using polyfit

```
%1 in polyfit stands for linear regression
p = polyfit(xvec,yvec1,1);
yfit = polyval(p,xvec);

%p(1) is the slope and p(2) is the intercept of the linear predictor
m = p(1);
nn = p(2);
fprintf('Slope (m) = %5.3f and intercept (n) = %5.3f.\n', m, nn);
```

Horizontalisation if needed

```
if m > 0.02 && m < -0.02
    fprintf('No need to make horizontal. \n');
    yvec2 = yvec1;
else
    ylinear = zeros(1,c);
    yvec2 = zeros(1,c);
    for i = 1:c
        ylinear(i) = m * xvec(i) + nn;
        yvec2(i) = yvec1(i) - ylinear(i);
    end
```

Verification of correctness in *horizontalisation*

```
    equaltozero = ismequaltozero(xvec, yvec2);

    if equaltozero
        fprintf('Incorrect horizontalisation!\n')
    else
        fprintf('Correct horizontalisation!\n')
        %create and save horizontal plot in htzmat
        htzmat = [xvec;yvec2];
        htzmat = transpose(htzmat);
    end

end
```

Calculation of Py and lc0; Print Table of parameters for yvec2

```
n=1;
[Py, lc] = calcry(xvec, yvec2, 1);
fprintf('Value of Py (or Pz) is Py = %1.3f micrometers.\n', Py);
fprintf('First Cut-off length is lcº = %0.2f mm according to Py. \n', lc);

[Pa, Pq] = calcrarq(xvec, yvec2, lc, 0);

Values = {'Primary Profile'};
T1 = table(Pa, Pq, Py, 'RowNames', Values);

fprintf('\n');
disp(name);
disp(T1);
roughclass(Pa);
```

Plot the data and label the plot

```
figure(1)
plot(xvec, yvec1, 'r', xvec, yvec2, 'g')
grid on
xlabel('Horizontal (mm)');
ylabel('Roughness (micras)');
title('Roughness of Surface');
legend(name, 'Primary profile');
axis([xmin xmax ymin-1 ymax+1]);
```

Iteration to find last value of lc

```
[yvecr, yvecw, hvec, lc] = cutfunc(lc, xvec, yvec2, name);
```

Plot data

```
pick = menu('Do you want all Data in one Plot?', 'Yes', 'No', 'Both things');
switch pick
    case 1
        figure(3)
        plot(xvec, yvec2, 'g', xvec, yvecw, 'r', xvec, yvecr, 'b');
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('All Data');
        legend('Primary Profile (yvec2)','Wave Profile (yvecw)','Roughness Profile
(yvecr)');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(4)
        plot(xvec, hvec)
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('Filter hvec');
    case 2
        figure(3)
        plot(xvec, yvec2, 'g');
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('Primary Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(4)
        plot(xvec, yvecw, 'r')
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('W Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(5)
        plot(xvec, yvecr, 'b')
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('R Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(6)
        plot(xvec,hvec)
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('Filter hvec');
    case 3
        figure(3)
        plot(xvec, yvec2, 'g', xvec, yvecw, 'r', xvec, yvecr, 'b');
        grid
```

```
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('All Data');
        legend('Primary Profile (yvec2)','Wave Profile (yvecw)','Roughness Profile
(yvecr)');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(4)
        plot(xvec, yvec2, 'g')
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('Primary Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(5)
        plot(xvec, yvecw, 'r')
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('W Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(6)
        plot(xvec, yvecr, 'b')
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('R Profile');
        axis([xmin xmax ymin-1 ymax+1]);

        figure(7)
        plot(xvec, hvec)
        grid
        xlabel('Horizontal (mm)');
        ylabel('Roughness (micrometers)');
        title('Filter hvec');
    otherwise
end
```

Save data

```
sdat = menu('Do you want to save Data?', 'Yes', 'No');
switch sdat
    case 1
        %create and save Horizontal plot in htzmat
        dlmwrite('Htzmat.txt',htzmat);
        disp('Horizontal profile values saved in htzmat.txt');
        %create and save W plot in wmat
        wmat = [xvec;yvecw];
        wmat = transpose(wmat);
        dlmwrite('Wmat.txt',wmat);
        disp('W profile values saved in Wmat.txt');
        %create and save R plot in rmat
        rmat = [xvec;yvecr];
```

```
        rmat = transpose(rmat);
        dlmwrite('Rmat.txt',rmat);
        disp('R profile values saved in Rmat.txt');
    case 2
        %Nothing done
        disp('Data not saved.');
    otherwise
end

fprintf('Value of final Cut-off length lc = %1.2f mm.\n', lc);
```

Save plots

```
spl = menu('Do you want to save Plots?', 'Yes', 'No');
switch spl
    case 1
        if pick == 1
            savefig(figure(3),'alldata.fig');
            savefig(figure(4),'hvec.fig');
        elseif pick == 2
            savefig(figure(3),'p profile.fig');
            savefig(figure(4), 'w profile.fig');
            savefig(figure(5),'r profile.fig');
            savefig(figure(6), 'hvec.fig');
        elseif pick == 3
            savefig(figure(3),'alldata.fig');
            savefig(figure(4),'p profile.fig');
            savefig(figure(5), 'w profile.fig');
            savefig(figure(6),'r profile.fig');
            savefig(figure(7), 'hvec.fig');
        else
        end
        disp('Plots saved.');
    case 2
        %Nothing done
        disp('Plots not saved.');
    otherwise
end
```

Ask for new lc

```
fin = menu('Do you want new profile or finish algorithm?', 'New profile', 'Finish');
switch fin
    case 1
        profiles;
    case 2
        %
    otherwise
end
```

Makevec.m

```
function [xvec, yvec1, baseFileName] = makevec()
%Convert data obtained in .txt to vectors
```

Get name of folder and file

```
startingFolder = 'C:\Users\Juan\Google Drive\PFC\.m\Data Converted\Bruto Mod';
if ~exist(startingFolder, 'dir')
    %If that folder doesn't exist, just start in the current folder.
    startingFolder = pwd;
end

% Get the name of the file that the user wants to use.
defaultFileName = fullfile(startingFolder, '*.*');
[baseFileName, folder] = uigetfile(defaultFileName, 'Select a file');
if baseFileName == 0
    % User clicked the Cancel button.
    return;
end
```

Open file

```
rugbruto = fullfile(folder, baseFileName);

fid = fopen(rugbruto);
if fid == -1
    warningMessage = sprintf('Error opening file:\n%s !', rugbruto);
    uiwait(warndlg(warningMessage));
    return;
else
    disp('File open successful!');
    disp(baseFileName);
end
```

Save on vectors

```
C = textscan(fid,'%f %f %f %f %u8 %f', 'Delimiter',',');
xvec = C{1};
yvec1 = C{2};
fclose(fid);

xvec = xvec';
yvec1 = yvec1';
end
```

Ismequaltozero.m

```
function equaltozero = ismequaltozero(xa,ya)

%Function that looks if plot is horizontal, that is to say, m = 0
%1 is true, 0 is false
```

Regression line of xa and ya

```
%p(1) is the slope and p(2) is the intercept of the linear predictor
pa = polyfit(xa,ya,1);
m = pa(1);
n = pa(2);

if abs(n) ~= 0
    %Continue
else
    n = abs(pa(2));
end
fprintf('New slope (m) = %5.3f and new intercept (n) = %5.3f\n', abs(m), abs(n));
```

Return

```
equaltozero = (m == 0.000);

end
```

Calcry.m

```
function [Ry, lc] = calcry(xvec, yvec, n)
```

```
%Calculus of Ry or Py and cut-off length, lc (lc0 for Py)

len = length(xvec)/n;
len = int16(fix(len));
yveclc = zeros(1, len);
Rysum = 0;
```

Determination of intervals and Ry

```
for i = 0:n-1
    for j = 1:len
        yveclc(j) = yvec(j+i*len);
    end
    Ry = abs(max(yveclc)) + abs(min(yveclc));
    Rysum = Rysum + Ry;
end

Ry = Rysum / n;

%[Ry] = micrometers
%[lc1] = mm
```

Tables

```
%According to ISO 4288-1997
if Ry > 0.025 && Ry <= 0.1
    lc = 0.08;
elseif Ry > 0.1 && Ry <= 0.5
    lc = 0.25;
elseif Ry > 0.5 && Ry <= 10
    lc = 0.8;
elseif Ry > 10 && Ry <= 50
    lc = 2.5;
elseif Ry > 50 && Ry <= 200
    lc = 8;
else
    disp('Wrong value for Ry');
    %Error
end
```

```
end
```

Calcrw.m

```
function [yvecr, yvecw, hvec] = calcrw (xvec, yvec, lc)
```

```
% Function that calculates yvecw and yvecr

len = length(xvec);
```

Gaussian filter vectors

```
hvec = zeros(1,len);
xvecesp = zeros(1,len);
A = xvec(2) - xvec(1);
a = 0.4697;

for i=1:length(xvec)
    xvecesp(i) = xvec(i) - A*(len/2); %To center the filter
    hvec(i) = exp(-pi * (xvecesp(i) / (a * lc))^2) / (a * lc);
end
hvec = hvec ./ max(hvec);
```

Convolution

```
%All matrices in the argument list must have the same number of columns.
%Returns only the central part of the convolution, the same size as hvec.
yvecw = conv(hvec, yvec, 'same');

%Correction of convolution units and proportion
yvecw = yvecw * max(yvec)/max(yvecw);
```

Obtaining roughness profile: yvecr = yvec - yvecw

```
num1 = max(yvec) + abs(min(yvec));
num2 = max(yvecw) + abs(min(yvecw));

num = max(num1, num2);

yvecaux = yvec + num;
yvecwaux = yvecw + num;

yvecr = yvecaux - yvecwaux;
```

```
end
```

Cutfunc.m

```
function [yvecr, yvecw, hvec, lc] = cutfunc(lc0, xvec, yvec2, name)
%Function that given a roughness plot, calculates the R and W profiles
```

Calculus via iterative function of final cut-off length, lc

```
ite = 1;
[lc, Ra, Rq] = calcite(lc0, xvec, yvec2, ite);
```

Calculus of final yvecw and yvecr

```
[yvecr, yvecw, hvec] = calcrw(xvec, yvec2, lc);
```

Print Table of parameters for yvecr

```
n = 1;
[Ry, ~] = calcry(xvec, yvecr, n);

Values = {'Parameter values R profile'};
T2 = table(lc, Ra, Rq, Ry, 'RowNames', Values);

fprintf('\n');
disp(name);
disp(T2);
roughclass(Ra);
```

```
end
```

Calcite.m

```
function [lc, Ra, Rq] = calcite(lc1, xvec, yvec2, ite)

%Calculus of cut-off length, lc, Ra and Rq
%Iterative function! (non periodic roughness profiles)
```

Cut-off, evaluation length and n

```
ln1 = lc1 * 5;
n = 5;

while ln1 >= max(xvec) - min(xvec)
    n = n - 1;
    ln1 = n * lc1;
end

fprintf('[IT(%1.0f)] Entered Cut-off length is lc = %1.2f mm, ln = %1.2f mm and n = %d.
\n', ite, lc1, ln1, n);
```

Obtaining Roughness Profile

```
[yvecr, ~, ~] = calcrw (xvec, yvec2, lc1);
```

Intervals of study

```
if n==0 | ln1==0 | lc1==0 %#ok<OR2>
    disp('Problem, n=0. ln > xmax.');
    lc2 = 5 / 6 * max(xvec);
    fprintf('Cut-off taken lc = %1.2f mm', lc2);
else

    [ints, intf] = calcint(ln1, xvec, ite);
    xvecln = zeros(1,intf-ints);
    yvecrln = zeros(1,intf-ints);
    j = 1;

    for i = ints:intf
        xvecln(j) = xvec(i);
        yvecrln(j) = yvecr(i);
        j = j + 1;
    end

    pa = polyfit(xvecln,yvecrln,1);
    g = polyval(pa, xvecln);

    %Verification of correctness of mean line
    sumneg = 0;
    sumpos = 0;
```

```
    i = 1;

    while i <= length(yvecrln)
        if yvecrln(i) <= g(i) && yvecrln(i) < 0
            sumneg = sumneg + g(i) + abs(yvecrln(i));
        elseif yvecrln(i) <= g(i) && yvecrln(i) >= 0
            sumneg = sumneg + (g(i)-yvecrln(i));
        elseif yvecrln(i) >= g(i)
            sumpos = sumpos + abs(yvecrln(i) - g(i));
        else
        end
        i = i + 1;
    end

    if sumneg >= 0.95*sumpos && sumneg <= 1.05*sumpos
        %disp('    Correct mean line!');
    else
        disp('   Incorrect mean line!');
        fprintf('   Sumpos = %2.2f & Sumneg = %2.2f\n', sumpos, sumneg);
    end
```

Calculus of Ra and new lc (lc2)

```
    [Ra, Rq] = calcrarq(xvecln, yvecrln, lc1, 0);
    [lc2] = valralc(Ra);

    n = 5;
    ln2 = lc2 * n; %Evaluation length and cut-off length
    while ln2 >= max(xvec)
        n = n - 1;
        ln2 = n * lc2;
    end

    fprintf('[IT(%1.0f)] Ra = %1.3f, new Cut-off length lc = %1.2f mm, ln = %1.2f mm and
n = %d. \n', ite, Ra, lc2, lc2*n, n);
```

Iteration

```
    if lc1 ~= lc2 || ite == 1
        ite = ite + 1;
        if ite > 4
            [Ra, Rq] = calcrarq(xvec, yvecr, lc1, n);
            [lc2] = valralc(Ra);
            fprintf('[IT(%1.0f)] Iteration finished: lc(%1.0f) = %1.2f. \n', ite, ite,
lc1);
        else
            [lc2, Ra, Rq] = calcite(lc1, xvec, yvec2, ite);
        end
    else
        fprintf('[IT(%1.0f)] Iteration finished: lc(%1.0f) = lc(%1.0f) = %1.2f. \n',
ite, ite-1, ite, lc1);
        %lc = lc1;
    end
```

```
end
lc = lc2;

end
```

Calcint.m

```
function [ints, intf] = calcint(ln, xvec, ite)
%Function that returns study interval start and finish values,
%ints and intf (interval start and interval finish)
```

It and lv

```
lt = xvec(length(xvec)) - xvec(1);
lv = 2/3*(lt-ln);
```

Indexes

```
i = 1;
while xvec(i)-xvec(1) <= lv
    i = i + 1;
end

ints = i-1;

while xvec(i)-xvec(1) <= lv+ln
    i = i + 1;
end

intf = i-1;
```

Plot

```
%figure(10+ite)
%grid on
%plot(xvec, yvec,'b');
%line([xvec(ints) xvec(ints)], [-5 10]);
%line([xvec(intf) xvec(intf)], [-5 10]);
%xlabel('Horizontal (mm)');
%ylabel('Roughness (micrometers)');
%title('Interval limits for yvec');

if ite ~= 0
    fprintf('[IT(%1.0f)] Value of limits for start = %0.0f, %1.3f mm, and for finish =
%0.0f, %1.2f mm.\n', ite, ints, xvec(ints), intf, xvec(intf));
end
```

```
end
```

Calcrarq.m

```
function [Ra, Rq] = calcrarq(xvec, yvec, lc, n)
```

```
%Calculus of Ra and Rq

ln = lc * n;

while ln >= max(xvec) - min(xvec)
    n = n - 1;
    ln = n * lc;
end
```

Obtaining limits for xvecln and yvecrln

```
%Distinction in values entered: if n == 0,
%values entered are xvecln and yvecrln
if n == 0
    xvecln = xvec;
    yvecrln = yvec;

else
    [ints, intf] = calcint(ln, xvec, 0);
    xvecln = zeros(1,intf-ints);
    yvecrln = zeros(1,intf-ints);
    j = 1;

    for i = ints:intf
        xvecln(j) = xvec(i);
        yvecrln(j) = yvec(i);
        j = j + 1;
    end
end
```

Main calculation of Ra and Rq

```
z = 0;
zsq = 0;
for i = 1:length(xvecln)
    z = abs(yvecrln(i)) + z;
    zsq = yvecrln(i)^2 + zsq;
end

Ra = z / length(xvecln);
Rq = sqrt(zsq/length(xvecln));
```

```
end
```

Valralc.m

```
function [lc] = valralc(Ra)
% Function giving value of lc for value of Ra according to tables

%Ra is in micrometers
%lc in mm

%According to ISO 4288 and ASME B46.1
if Ra > 0.006 && Ra <= 0.002
    lc = 0.08;
elseif Ra > 0.002 && Ra <= 0.1
    lc = 0.25;
elseif Ra > 0.1 && Ra <= 2
    lc = 0.8;
elseif Ra > 2 && Ra <= 10
    lc = 2.5;
elseif Ra > 10 && Ra <= 80
    lc = 8;
else
    disp('ERROR: Wrong value for Ra! Ra either <0.002 or >80.');
    lc = 0.08;
    %Error
end

end
```

Roughclass.m

```
function rc = roughclass(Ra)
%Display of roughness class

%According to unreferenced sources
if Ra >= 50
    rc = 12;
elseif Ra < 50 && Ra >= 25
    rc = 11;
elseif Ra < 25 && Ra >= 12.5
    rc = 10;
elseif Ra < 12.5 && Ra >= 6.3
    rc = 9;
elseif Ra < 6.3 && Ra >= 3.2
    rc = 8;
elseif Ra < 3.2 && Ra >= 1.6
    rc = 7;
elseif Ra < 1.6 && Ra >= 0.8
    rc = 6;
elseif Ra < 0.8 && Ra >= 0.4
    rc = 5;
elseif Ra < 0.4 && Ra >= 0.2
    rc = 4;
elseif Ra < 0.2 && Ra >= 0.1
    rc = 3;
elseif Ra < 0.1 && Ra >= 0.05
    rc = 2;
elseif Ra < 0.05 && Ra >= 0.025
    rc = 1;
else
    rc = 0;
end

fprintf('- The class of roughness is N%d\n', rc);

end
```