APPENDIX

Aspects of solid-state chemistry of fly ash and ultramarine pigments
A1. SPECTRAL MANIPULATIONS PROGRAM

This program incorporates previous programs and allows the user to manipulate spectra in several ways and calculate relative areas as defined by the user.

The area is calculated by the trapezoidal Rule as described in


uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, math;

type
 TForm1 = class(TForm)
  CloseButton: TBtn;
  Bevel1: TBevel;
  intro_Label: TLabel;
  infiles_Label: TLabel;
  wavenumber_Label: TLabel;
  intensity_Label: TLabel;
  baseline_Label: TLabel;
  outfiles_Label: TLabel;
  outwavenumber_Label: TLabel;

  procedure TForm1.TBtnClick(Sender: TObject);
end;

unit Spectra_an;

{This program incorporates previous programs and allows the user to manipulate spectra in several ways and calculate relative areas as defined by the user.}

interface

{The area is calculated by the trapezoidal Rule as described in

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, math;

type
TForm1 = class(TForm)
  CloseButton: TBtn;
  Bevel1: TBevel;
  intro_Label: TLabel;
  infiles_Label: TLabel;
  wavenumber_Label: TLabel;
  intensity_Label: TLabel;
  baseline_Label: TLabel;
  outfiles_Label: TLabel;
  outwavenumber_Label: TLabel;

  procedure TForm1.TBtnClick(Sender: TObject);
end;
outintensity_Label: TLabel;
added_programmes_Label: TLabel;
temp1_Label: TLabel;
temp2_Label: TLabel;
sequential_Label: TLabel;
CalculateButton: TButton;
endbox: TEdit;
areaout_Label: TLabel;
Norm_int_radio: TRadioButton;
numberofspectrainput: TEdit;
Normarea_radio: TRadioButton;
Norm_area_input: TEdit;
Base_corr_radio: TRadioButton;
base_int_corr_radio: TRadioButton;
base_area_corr_radio: TRadioButton;
norm_area_input2: TEdit;
standard_area_calc_radio: TRadioButton;
costum_areaCalc_radio: TRadioButton;
region1_CheckBox: TCheckBox;
region1begin: TEdit;
region1end: TEdit;
region2_CheckBox: TCheckBox;
region2begin: TEdit;
region2end: TEdit;
region3_CheckBox: TCheckBox;
region3begin: TEdit;
region3end: TEdit;
region4_CheckBox: TCheckBox;
region4begin: TEdit;
region4end: TEdit;
region5_CheckBox: TCheckBox;
region5begin: TEdit;
region5end: TEdit;
region6_CheckBox: TCheckBox;
region6begin: TEdit;
region6end: TEdit;
calc_ratio_Label: TLabel;
numerateregion: TEdit;
ratio_to_Label: TLabel;
ratiodenominator: TEdit;
MaximumvalueCheckBox: TCheckBox;
maxwaveinput: TEdit;
othermaxint_radio: TRadioButton;
input_maxint: TEdit;
input_maxint2: TEdit;
procedure CalculateButtonClick(Sender: TObject);
end;

var
Form1: TForm1;
datasize, topratioarea, bottomratioarea: integer;
maximumwavelength, maxintensity: real;
total, normarea, topratioareareal, bottomratioareareal: real;
intensityarray, basearray, areaarray: array of array of real;
totalint, flag, areaarraydimension: integer;
I, J, A, B, err, temp: integer;
interval: array[0..11] of integer;
wavearray, minarray, maxarray: array of real;
namearray: array of string;
textfile, textfile2: text;
region1min, region1max, region2min, region2max: real;
region3min, region3max, region4min, region4max: real;
region5min, region5max, region6min, region6max: real;
inputerror: boolean;
inputused: array[1..6] of boolean;
implementation
{$R *.DFM}
Procedure wavenumbers_size_maximumset; {Determine the number of data points that the user is interested in}
var
x: integer;
wavenumber: real;
begin
assignfile (textfile,'wavenumber.dat');
reset(textfile);
x:= 0;
readln(textfile,wavenumber);
x:= x + 1;
while ((wavenumber < maximumwavelength) and not(eof(textfile))) do begin
readln(textfile,wavenumber);
x:= x + 1;
end;
if wavenumber > maximumwavelength
then datasize:= x-2
else datasize:= x-1;
closefile(textfile);
end;

Procedure wavenumbers_size_fileset; {Determine the number of data points that the file supplies}
var
  x: integer;
  wavenumber: real;
begin
  assignfile (textfile,'wavenumber.dat');
  reset(textfile);
  x:= 0;
  readln(textfile,wavenumber);
  x:= x + 1;
  while not(eof(textfile)) do
  begin
    readln(textfile,wavenumber);
    x:= x + 1;
  end;
  datasize:= x;
  closefile(textfile);
end;

Procedure initialise_arrays; {Specify the size of the arrays}
begin
  setlength(intensityarray, totalint, datasize);
  setlength(wavearray, datasize);
  setlength(minarray, totalint);
  setlength(areaarray, totalint, areaarraydimension);
  setlength(maxarray, totalint);
  setlength(namearray, totalint);
end;

Procedure assign_wavenumbers; {Assign values to the wavenumber array}
var
  x: integer;
begin
  assignfile (textfile,'wavenumber.dat');
  reset(textfile);
  for x:= 0 to (datasize-1) do
  begin
    readln(textfile, wavearray[x]);
  end;
  closefile(textfile);
end;

Procedure intensities; {Input the absorbance values}
var
  x, y, position: integer;
  tempstring: string;
begin
  assignfile (textfile,'outintensity.prn');
  reset(textfile);
  readln(textfile, tempstring);
  for x:= 0 to (totalint-2) do
  begin
    position:= pos(' ',tempstring);
    if position = 1
    then position:= pos(' ',copy(tempstring,2,1000000));
    namearray[x]:= copy(tempstring, 1, position);
    delete(tempstring,1,position);
    namearray[totalint-1]:= tempstring;
  end;
  for x:= 0 to (datasize-1) do
  begin
    Appendix A4
Aspects of solid-state chemistry of fly ash and ultramarine pigments

for y:= 0 to (totalint - 1) do
    begin
        read(textfile,intensityarray[y,x]);
        end;
    end;
closefile(textfile);

Procedure baseline; {Input Baseline parameters}
var x, y: integer;
begin
    assignfile (textfile,'baseline.dat');
    reset(textfile);
    readln(textfile, tempstring);
    for x:= 0 to 3 do
        begin
            for y:= 0 to (totalint - 1) do
                begin
                    read(textfile,basearray[y,x]);
                    end;
            end;
closefile(textfile);
end;

Procedure find_min; {Find minimum absorbance}
var x, y: integer;
begin
    for x:= 0 to (totalint -1) do
        begin
            minarray[x]:= intensityarray[x,0];
            for y:= 1 to (datasize-1) do
                begin
                    minarray[x]:= min(minarray[x],intensityarray[x,y]);
                    end;
        end;

Procedure find_max; {Find maximum absorbance}
var x, y: integer;
begin
    for x:= 0 to (totalint -1) do
        begin
            maxarray[x]:= intensityarray[x,0];
            for y:= 1 to (datasize-1) do
                begin
                    maxarray[x]:= max(maxarray[x],intensityarray[x,y]);
                    end;
        end;

Procedure normalise_intensity; {Change intensities so that all absorbance values are between 0 and 1}
var x, y: integer;
begin
    for x:= 0 to (totalint - 1) do
        begin
            for y:= 0 to (datasize-1) do
                begin
                    intensityarray[x,y]:= (intensityarray[x,y]-minarray[x])/(maxarray[x]-minarray[x])*maxintensity;
                end;
        end;
procedure calculate_total_area; {Calculate the current total area}
var x, y: integer;
beg
for x:= 0 to (totalint - 1) do
begin
areaarray[x,0] := 0;
for y:= 0 to (datasize - 2) do
begin
areaarray[x,0] := areaarray[x,0] - 0.5*(wavearray[y]-wavearray[y+1])*(intensityarray[x,y]+intensityarray[x,y+1]-2*minarray[x]);
end;
end;
end;

Procedure normalise_area; {Insure that the minimum absorbance is 0 and that the total area
equals the desired normarea}
var x, y: integer;
beg
for x:= 0 to (totalint - 1) do
begin
for y:= 0 to (datasize-1) do
begin
intensityarray[x,y] := normarea*((intensityarray[x,y]-minarray[x])/(areaarray[x,0]));
end;
end;
end;

Procedure baseline_correction; {Subtract the baseline}
var x, y: integer;
beg
for x:= 0 to (totalint-1) do
begin
for y:= 0 to (datasize-1) do
begin
intensityarray[x,y] := intensityarray[x,y] - basearray[x,0]*(wavearray[y]*wavearray[y]*wavearray[y]) - basearray[x,1]*(wavearray[y]*wavearray[y])- basearray[x,2]*wavearray[y]- basearray[x,3];
end;
end;
end;

Procedure determine_intervals; {The starting wavenumbers for the six regions are defined}
var x: integer;
beg
flag:= 0;
For x:= 0 to (datasize-1) do
begin
if (wavearray[x] >= region1min) and (flag = 0) then
begin
interval[flag]:= x;
flag:= flag + 1;
end;
end;
if (wavearray[x] >= region2min) and (flag = 1) then
begin
interval[flag]:= x;
flag:= flag + 1;
end;
if (wavearray[x] >= region3min) and (flag = 2) then
begin
interval[flag]:= x;
flag:= flag + 1;
end;
end;

begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region4min) and (flag = 6)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region4max) and (flag = 7)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region5min) and (flag = 8)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region5max) and (flag = 9)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region6min) and (flag = 10)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
if (wavearray[x] >= region6max) and (flag = 11)
then
begin
  interval[flag]:= x;
  flag:= flag + 1;
end;
end;

Procedure write_output; {Export the intensity normalised spectra}
var x, y, z: integer;
begin
  Assignfile (textfile,'outintensity.prn');
  Assignfile (textfile2, 'outwavenumber.dat');
  rewrite(textfile);
  rewrite(textfile2);
  for z:= 0 to (totalint -1) do
  begin
    write(textfile, namearray[z]);
    write(textfile, ' ');
  end;
  writeln(textfile);
  for x:= 0 to (datasize-1) do
  begin
    for y:= 0 to (totalint-1) do
    begin
      write(textfile, intensityarray[y,x]);
      write(textfile, ' ');
    end;
    writeln(textfile);
    writeln(textfile2, wavearray[x]);
  end;
  closefile(textfile);
  closefile(textfile2);
end;

Procedure Mean Frequency; {Calculate the mean frequency}
var x, y, flag: integer;
begin
  for x:= 0 to (totalint-1) do
  begin
    flag:= 0;
    y:= 0;
    area:= 0;
    while flag = 0 do
    begin
      area:= area - 0.5*(wavearray[y]-wavearray[y+1])*
      (intensityarray[x,y]+intensityarray[x,y+1]-2*minarray[x]);
      if (area > (0.5*areaarray[x,0]))
      then
      begin
        areaarray[x,8]:= wavearray[y];
        flag:= flag +1;
      end;
      y:= y + 1;
    end;
  end;
procedure relative_areas; {Calculate the relative area in each region}
var x, y, n: integer;
begin
  for x:= 0 to (totalint - 1) do
    begin
      n:= 0;
      while n <= 11 do
        begin
          y:= interval[n];
          while y <= interval[n+1] do
            begin
              areaarray[x,round((n/2)+1)]:= areaarray[x,round((n/2)+1)]-0.5*(wavearray[y]-wavearray[y+1])*(intensityarray[x,y]+intensityarray[x,y+1] - 2*minarray[x]);
              y:= y + 1;
            end;
          areaarray[x,round((n/2)+1)]:= areaarray[x,round((n/2)+1)]/areaarray[x,0]*100;
          n:= n + 2;
        end;
      areaarray[x,7]:= areaarray[x,topratioarea]/areaarray[x,bottomratioarea]; {ratio}
    end;
end;

Procedure write_relative_area_output; {Export the Area, ratio and mean frequency data}
var x, y: integer;
begin
  Assignfile (textfile,'areaoutput.dat');
  rewrite(textfile);
  writeln(textfile, 'Name Total Region1 Region2 Region3 Region4 Region5 Region6 Ratio Mean Frequency');
  for x:= 0 to (totalint - 1) do
    begin
      write(textfile, namearray[x]);
      for y:= 0 to 8 do
        begin
          write(textfile, areaarray[x,y]);
        end;
      writeln(textfile,'');
    end;
  closefile(textfile);
end;

procedure TForm1.CalculateButtonClick(Sender: TObject);
begin
  inputerror:= false;
  {Obtain the dimensions of the arrays based on wavenumber}
  if MaximumvalueCheckBox.Checked
    then
      begin
        val (maxwaveinput.text, maximumwavelength,err);
        wavenumbers_size_maximumset;
      end;
  if standard_area_calc_radio.Checked
    then
      begin
        maximumwavelength:= 2500;
        wavenumbers_size_maximumset;
        {define interval regions}
      end;
  if not(MaximumvalueCheckBox.Checked or standard_area_calc_radio.Checked)
    then wavenumbers_size_fileset;
  {Obtain the dimensions of the arrays based on number of spectra}
  Val (numberofspectrainput.text,total,err);
  totalint:= round(total);
  {Obtain the dimensions of the arrays based on whether the areaarray is used to normalise the area or do a region analysis}
if (costum_area_calc_radio.Checked or standard_area_calc_radio.Checked)
then areaarraydimension:= 9
else areaarraydimension:= 1;

{Input intensities into initialised arrays}
initialise_arrays;
assign_wavenumbers;
fix_minus_intensity;

get_rid_of_extra_spaces_intensity;
inintensities;

{Normalise intensity to 1}
if Norm_int_radio.Checked
then
begin
maxintensity:= 1;
find_min;

{Normalise intensity to another value}
if othermaxint_radio.Checked
then
begin
val
(input_maxint.text,maxintensity,err);
find_min;

{Area normalisation}
if normarea_radio.Checked
then
begin
val
(Norm_area_input.text,normarea,err);
find_min;

{Baseline correction and area normalisation}
if base_area_corr_radio.Checked
then
begin
fix_minus_base;
get_rid_of_extra_spaces_base;
setlength (basearray, totalint, 4);
baseline;
baseline_correction;

val
(Norm_area_input2.text,normarea,err);
find_min;
find_max;
calculate_total_area;
normalise_area;
write_output;
endbox.text:= 'Done';
end;

{Baseline correction without area normalisation}
if Base_corr_radio.Checked
then
begin
fix_minus_base;
get_rid_of_extra_spaces_base;
setlength (basearray, totalint, 4);
baseline;
baseline_correction;

endbox.text:= 'Done';
end;

{Baseline correction and intensity normalisation}
if base_int_corr_radio.Checked
then
begin
fix_minus_base;
get_rid_of_extra_spaces_base;
setlength (basearray, totalint, 4);
baseline;
baseline_correction;

val
(input_maxint2.text,maxintensity,err);
find_min;
find_max;
normalise_intensity;
write_output;
endbox.text:= 'Done';
end;

{Calculate predefined area ratios}
if standard_area_calc_radio.Checked
then
begin
region1min:= 401;
region1max:= 500;
region2min:= 501;
region2max:= 800;
region3min:= 801;
region3max:= 1000;
region4min:= 1001;
region4max:= 1300;
region5min:= 1301;
region5max:= 1700;
region6min:= 1701;
region6max:= 2498;
topratioarea:= 2;
bottomratioarea:= 1;
determine_intervals;
find_min;

{Calculate user defined area ratios}
if costum_area_calc_radio.Checked
   then
      begin
         if region1_CheckBox.Checked
            then
               begin
                  val (region1begin.text, region1min, err);
                  val (region1end.text, region1max, err);
                  if (region1min >= region1max)
                     or (region1min < wavearray[0])
                     or (region1max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[1]:= true;
               end
            else
               begin
                  region1min:= wavearray[1];
                  region1max:= wavearray[0];
                  inputused[1]:= false;
               end;
         if region2_CheckBox.Checked
            then
               begin
                  val (region2begin.text, region2min, err);
                  val (region2end.text, region2max, err);
                  if (region2min >= region2max)
                     or (region2min < wavearray[0])
                     or (region2max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[2]:= true;
               end
            else
               begin
                  region2min:= wavearray[1];
                  region2max:= wavearray[0];
                  inputused[2]:= false;
               end;
         if region3_CheckBox.Checked
            then
               begin
                  val (region3begin.text, region3min, err);
                  val (region3end.text, region3max, err);
                  if (region3min >= region3max)
                     or (region3min < wavearray[0])
                     or (region3max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[3]:= true;
               end
            else
               begin
                  region3min:= wavearray[1];
                  region3max:= wavearray[0];
                  inputused[3]:= false;
               end;
         if region4_CheckBox.Checked
            then
               begin
                  val (region4begin.text, region4min, err);
                  val (region4end.text, region4max, err);
                  if (region4min >= region4max)
                     or (region4min < wavearray[0])
                     or (region4max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[4]:= true;
               end
            else
               begin
                  region4min:= wavearray[1];
                  region4max:= wavearray[0];
                  inputused[4]:= false;
               end;
         if region5_CheckBox.Checked
            then
               begin
                  val (region5begin.text, region5min, err);
                  val (region5end.text, region5max, err);
                  if (region5min >= region5max)
                     or (region5min < wavearray[0])
                     or (region5max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[5]:= true;
               end
            else
               begin
                  region5min:= wavearray[1];
                  region5max:= wavearray[0];
                  inputused[5]:= false;
               end;
         if region6_CheckBox.Checked
            then
               begin
                  val (region6begin.text, region6min, err);
                  val (region6end.text, region6max, err);
                  if (region6min >= region6max)
                     or (region6min < wavearray[0])
                     or (region6max > wavearray[datasize-1])
                     then inputerror:= true;
                  inputused[6]:= true;
               end
            else
               begin
                  region6min:= wavearray[1];
                  region6max:= wavearray[0];
                  inputused[6]:= false;
               end;
      end
end
end

{Determine intervals}
end
end

{Calculate total area}
calculate_total_area;
relative_areas;
Mean Frequency;
write_relative_area_output;
endbox.text:= 'Done';
end;
A2. VOLUME CALCULATION PROGRAM

{This program uses the Cartesian coordinates and atomic numbers of the input file to calculate the volume based on the method of A Gavezzotti, J. Am. Chem. Soc. 105, 5220-5225, 1983
The atomic radii are taken from Gavezzotti, R.D. Shannon, Acta Cryst. A32, 751-767, 1976

unit Vol_calc;
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Buttons, Math;
type
TForm1 = class(TForm)
  CloseBitBtn: TBitBtn;
  Calculate_button: TButton;
  files_GroupBox: TGroupBox;
  input_Label: TLabel;
  format_Label: TLabel;
  description_Label: TLabel;
  example_Label: TLabel;
end;
var

Appendix A11
Form1: TForm1;
coordinate_array: array of array of real;
In_atom_array: array of array of array of real;
VDWrad, Atom_rad: array of real;
number_of_atoms, err, number_pointsx, number_pointsy, number_pointsz: integer;
volume_total, volume_incr, startx, starty, startz: real;
volume_atoms, volume_cavity, endx, endy, endz: real;
flag_inside_atom, flag_in_cavity: boolean;

implementation
{$R *.DFM}

procedure input_vdwrad;
var
textfile: text;
x: integer;
{Reads the van der Waals radii into the array}
begi
assignfile (textfile, 'vdw_rad.txt');
reset(textfile);
setlength(VDWrad, 103);
for x:= 0 to 102 do
begin
readln(textfile);
readln(textfile, VDWrad[x]);
end;
end;

procedure input_from_file;
var
textfile: text;
x, y: integer;
Atomnumber_real: real;
{Reads in the data from the input file: input.prn}
begi
assignfile (textfile, 'input.prn');
reset(textfile);
{First determine the number of atoms}
x:= 0;
while not(eof(textfile)) do
begin
readln(textfile);
x:= x + 1;
end;
number_of_atoms:= x-1;
reset(textfile);
{Initialise the dynamic arrays}
setlength(coordinate_array, (number_of_atoms+1), 3);
setlength(Atom_rad, (number_of_atoms+1));
{Read the values into the appropriate arrays}
for x:= 0 to number_of_atoms do
begin
read(textfile, Atomnumber_real);
Atom_rad[x]:= VDWrad[round(Atomnumber_real)-1];
for y:= 0 to 2 do
begin
readln(textfile); coordinate_array[x, y];
end;
readln(textfile);
end;
closefile(textfile);
end;

procedure coordinate_system;
var
x: integer;
maxrad: real;
{Determines the dimensions of the volume to be searched}
begi;
Aspects of solid-state chemistry of fly ash and ultramarine pigments

for x:= 0 to number_of_atoms do
begin
endx:= max(endx,coordinate_array[x,0]);
endy:= max(endy,coordinate_array[x,1]);
endz:= max(endz,coordinate_array[x,2]);
startx:= min(startx,coordinate_array[x,0]);
starty:= min(starty,coordinate_array[x,1]);
startz:= min(startz,coordinate_array[x,2]);
maxrad:= max(maxrad,atom_rad[x]);
end;

{ensure that the molecule is in the correct octant of 3D space}
for x:= 0 to number_of_atoms do
begin
coordinate_array[x,0]:= coordinate_array[x,0] + abs(startx) + 1.1*maxrad;
coordinate_array[x,1]:= coordinate_array[x,1] + abs(starty) + 1.1*maxrad;
coordinate_array[x,2]:= coordinate_array[x,2] + abs(startz) + 1.1*maxrad;
end;

number_pointsx:= trunc((2*1.1*maxrad + endx - startx)/volume_incr) + 1;
number_pointsy:= trunc((2*1.1*maxrad + endy - starty)/volume_incr) + 1;
number_pointsz:= trunc((2*1.1*maxrad + endz - startz)/volume_incr) + 1;
setlength(in_atom_array, number_pointsx+1, number_pointsy+1, number_pointsz+1);

Procedure initialise_in_atom_array;
var
x, y, z:integer;
begin
for x:= 0 to number_pointsx do
begin
end;

procedure calculate_volume;
var
a, x, y, z, molendz:integer;
posx, posy, posz, centex, centery, centerz, sum_total, sum_cavity, sum_atoms : real;
flag_inside_mol, flag_inside_wall: boolean;
begin;
initialise_in_atom_array;
for x:= 0 to number_pointsx do
begin
posx:= x*volume_incr;
for y:= 0 to number_pointsy do
begin
posy:= y*volume_incr;
for z:= 0 to number_pointsz do
begin
posz:= z*volume_incr;
flag_inside_atom:= false;
a:= 0;
while ((a <= number_of_atoms) and not(flag_inside_atom)) do

begin
  centerx:= posx - coordinate_array[a,0];
  centery:= posy - coordinate_array[a,1];
  centerz:= posz - coordinate_array[a,2];
  flag_inside_atom:= (((centerx*centerx)+(centery*centery)+(centerz*centerz))
  <= (Atom_rad[a]*Atom_rad[a]));
  if flag_inside_atom
    then in_atom_array[x,y,z]:= 1;
    a:= a +1;
  end;
end;
end;
end;
sum_total:= 0;
sum_atoms:= 0;
sum_cavity:= 0;
for x:= 0 to number_pointsx do
  begin
    for y:= 0 to number_pointsy do
      begin
        flag_inside_mol:= false;
        flag_inside_wall:= false;
        molendz:= number_pointsz;
        z:= number_pointsz;
        while z >= 0 do
          begin
            if (in_atom_array[x,y,z] = 1)
              then
                begin
                  molendz:= z;
                  z:= 0;
                end;
            z:= z -1;
          end;
        for z:= 1 to molendz do
          begin
            sum_atoms:= sum_atoms + in_atom_array[x,y,z];
            if (in_atom_array[x,y,z] = 1)
              then
                begin
                  flag_inside_mol:= true;
                  flag_inside_wall:= true;
                end;
            if flag_inside_mol and not (in_atom_array[x,y,z] = in_atom_array[x,y,(z-1)])
              then flag_inside_wall:= false;
            if flag_inside_mol and not(flag_inside_wall)
              then sum_cavity:= sum_cavity + 1;
            if flag_inside_mol
              then sum_total:= sum_total + 1;
          end;
        end;
      end;
    volume_incr:= Power(volume_incr,3);
  volume_total:= sum_total*volume_incr;
  volume_cavity:= sum_cavity*volume_incr;
  volume_atoms:= sum_atoms*volume_incr;
end;
procedure output;
var
  x, y, z:integer;
  textfile: text;
{The results are saved in the file output.prn}
begin
  assignfile (textfile,'output.prn');
  rewrite(textfile);
  write(textfile, 'Total volume: ');
  writeln(textfile,volume_total);
  write(textfile, 'Volume occupied by the atoms: ');
  writeln(textfile,volume_atoms);
  write(textfile, 'Cavity volume: ');
  writeln(textfile,volume_cavity);
  closefile(textfile);
end;

procedure TForm1.Calculate_buttonClick(Sender: TObject);
begin;
  input_vdwrad;
  input_from_file;
  Val(Presicion.text,volume_incr,err);
  coordinate_system;
  calculate_volume;
  output;
  user_comm.text:= 'Done';
end;
end.

B. RAW DATA

B1. ULTRAMARINE BLUE KINETIC PEAK FITTING DATA

<table>
<thead>
<tr>
<th>Spectrum</th>
<th>Peak 1</th>
<th>Peak 1 Int</th>
<th>Peak 2</th>
<th>Peak 2 Int</th>
<th>Ratio</th>
<th>Ave Ratio</th>
<th>Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>umbl10</td>
<td>542.725</td>
<td>3456.36</td>
<td>578.986</td>
<td>670.653</td>
<td>0.2</td>
<td>0.19</td>
<td>0.02</td>
</tr>
<tr>
<td>umbl102</td>
<td>540.069</td>
<td>2283.55</td>
<td>574.995</td>
<td>958.82</td>
<td>0.4</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>umbl103</td>
<td>543.762</td>
<td>1183.07</td>
<td>578.865</td>
<td>215.091</td>
<td>0.2</td>
<td>0.15</td>
<td>0.04</td>
</tr>
<tr>
<td>umbl104</td>
<td>541.757</td>
<td>1197.34</td>
<td>576.442</td>
<td>463.313</td>
<td>0.4</td>
<td>0.15</td>
<td>0.04</td>
</tr>
<tr>
<td>umbl301</td>
<td>546.424</td>
<td>193.663</td>
<td>585.666</td>
<td>21.9575</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl302</td>
<td>545.009</td>
<td>705.423</td>
<td>582.165</td>
<td>81.9958</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl303</td>
<td>544.169</td>
<td>1854.61</td>
<td>580.265</td>
<td>287.992</td>
<td>0.2</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl304</td>
<td>542.403</td>
<td>4272.75</td>
<td>578.351</td>
<td>857.254</td>
<td>0.2</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl305</td>
<td>543.625</td>
<td>1178.56</td>
<td>579.748</td>
<td>222.864</td>
<td>0.2</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl60</td>
<td>544.527</td>
<td>2808.5</td>
<td>582.919</td>
<td>347.586</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl602</td>
<td>545.156</td>
<td>2785.09</td>
<td>583.421</td>
<td>314.326</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl603</td>
<td>544.237</td>
<td>4318.1</td>
<td>581.031</td>
<td>578.612</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl604</td>
<td>544.215</td>
<td>6725.48</td>
<td>581.998</td>
<td>851.471</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl95</td>
<td>545.967</td>
<td>759.234</td>
<td>578.029</td>
<td>69.1982</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl953</td>
<td>546.469</td>
<td>1253.29</td>
<td>583.098</td>
<td>93.691</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl954</td>
<td>545.479</td>
<td>1483.57</td>
<td>582.583</td>
<td>157.149</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>umbl955</td>
<td>544.44</td>
<td>4829.83</td>
<td>578.315</td>
<td>617.86</td>
<td>0.1</td>
<td>0.12</td>
<td>0.01</td>
</tr>
<tr>
<td>Umblack</td>
<td>543.982</td>
<td>1506.58</td>
<td>579.563</td>
<td>263.521</td>
<td>0.2</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>Umblack2</td>
<td>543.945</td>
<td>3788.4</td>
<td>577.968</td>
<td>900.849</td>
<td>0.2</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>Umblack3</td>
<td>543.943</td>
<td>1146.22</td>
<td>578.36</td>
<td>2456.75</td>
<td>0.2</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>Umblack4</td>
<td>542.649</td>
<td>3574.18</td>
<td>579.635</td>
<td>853.635</td>
<td>0.2</td>
<td>0.22</td>
<td>0.03</td>
</tr>
</tbody>
</table>
C. RAW DATA FOR THE CALCULATION OF THE TRUNCATED OCTAHEDRON VOLUME IN SODALITE CRYSTALS

<table>
<thead>
<tr>
<th>Authors</th>
<th>Large Base Å</th>
<th>Small Base Å</th>
<th>Diagonal Height Å</th>
<th>Height Å</th>
<th>Volume Å³</th>
</tr>
</thead>
<tbody>
<tr>
<td>Werner1</td>
<td>6.273</td>
<td>4.435</td>
<td>4.959</td>
<td>3.841</td>
<td>217.1</td>
</tr>
<tr>
<td>Werner2</td>
<td>6.202</td>
<td>4.386</td>
<td>4.903</td>
<td>3.797</td>
<td>209.8</td>
</tr>
<tr>
<td>Werner3</td>
<td>6.156</td>
<td>4.353</td>
<td>4.867</td>
<td>3.770</td>
<td>205.2</td>
</tr>
<tr>
<td>Werner4</td>
<td>6.092</td>
<td>4.308</td>
<td>4.816</td>
<td>3.730</td>
<td>198.9</td>
</tr>
<tr>
<td>Flesche</td>
<td>6.234</td>
<td>4.408</td>
<td>4.928</td>
<td>3.817</td>
<td>213.1</td>
</tr>
<tr>
<td>Nilesen1</td>
<td>6.276</td>
<td>4.438</td>
<td>4.962</td>
<td>3.844</td>
<td>217.5</td>
</tr>
<tr>
<td>Nilesen2</td>
<td>6.316</td>
<td>4.466</td>
<td>4.993</td>
<td>3.867</td>
<td>221.6</td>
</tr>
<tr>
<td>Nilesen3</td>
<td>6.370</td>
<td>4.505</td>
<td>5.036</td>
<td>3.900</td>
<td>227.4</td>
</tr>
<tr>
<td>Mead1</td>
<td>6.362</td>
<td>4.498</td>
<td>5.029</td>
<td>3.895</td>
<td>226.5</td>
</tr>
<tr>
<td>Mead2</td>
<td>6.380</td>
<td>4.511</td>
<td>5.043</td>
<td>3.906</td>
<td>228.4</td>
</tr>
<tr>
<td>Mead3</td>
<td>6.363</td>
<td>4.500</td>
<td>5.030</td>
<td>3.896</td>
<td>226.6</td>
</tr>
<tr>
<td>Mead4</td>
<td>6.394</td>
<td>4.522</td>
<td>5.056</td>
<td>3.916</td>
<td>230.0</td>
</tr>
<tr>
<td>Lons</td>
<td>6.272</td>
<td>4.435</td>
<td>4.958</td>
<td>3.840</td>
<td>217.0</td>
</tr>
</tbody>
</table>
Aspects of solid-state chemistry of fly ash and ultramarine pigments

D. POSTERS


Practical interpretation of infrared spectra of silicates

Andreas A Landman1 and Danita de Waal2*

1. Landman@science.up.ac.za 2. DeWaal@science.up.ac.za

Introduction

- Introduction to qualitative analysis with a statement roughly meaning: "In each of such a study, the concept of solid-state chemistry is a major tool in the analysis of silicate systems, and the interpretation of infrared spectra provides valuable insights into the structural and compositional aspects of these materials."

Definitions and concepts

- The different requirements and restrictions need to be considered when interpreting infrared spectra of silicates. It is important to note that the absorption bands observed in the spectra are related to the vibrational modes of the silicate framework, and these modes are influenced by the chemical environment and the structural arrangement of the silicate network.

Experimental details

- The procedure involved in the collection of infrared spectra involved the use of a Fourier transform infrared (FTIR) spectrometer. The samples were prepared as KBr pellets for transmission measurements.

Conclusions from literature

- A comprehensive literature review was conducted to identify key features of the infrared spectra of silicates. The following conclusions were drawn:

Visual interpretation of the infrared spectra

- The infrared spectra of silicates were interpreted using the information from the literature review. The key features of the spectra were identified and correlated with the structural and compositional characteristics of the silicate samples.

Numerical interpretation of the infrared spectra

- The absorbance values of the infrared spectra were analyzed using numerical methods to identify the most significant vibrational modes. The results were compared with those obtained from previous studies, and the consistency of the observations was discussed.

Table 1: Calculated relative absorbances of silicate species

<table>
<thead>
<tr>
<th>Silicate Species</th>
<th>Relative Absorbance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fly ash</td>
<td>1.000</td>
</tr>
<tr>
<td>Fly ash + Na2O</td>
<td>0.850</td>
</tr>
<tr>
<td>Fly ash + 100°C</td>
<td>0.750</td>
</tr>
</tbody>
</table>

Further notes

- The theoretical framework provided in the study can be extended to other silicate systems, and the methodology adopted in the analysis can be applied to similar studies.

Acknowledgements

- The authors acknowledge the financial support of the National Research Foundation (NRF) and the University of Pretoria for their contributions to this research.

References


Appendix A17
The Nature of the Ultramarine Red Chromophore

Andreas A. Landman, Jan CA Boeyens, Danita de Waal

University of Pretoria, Department of Chemistry, 0002, Pretoria, South Africa.
E-mail: alandman@postحو. up. za, ddewaal@postحو. up. za, jboeyens@postحو. up. za

The Nature of the Ultramarine Red Chromophore

Known facts about Ultramarine Red

Ultramarine pigments come in a range of shades and include blue, green, blue-green, green-blue, and blue-green. The blue is made by case-pairing with a mixture of blue-green, green-blue, and blue-green. The blue is also made by case-pairing, but with an additional amount of white.

The Structures Modelled and the Method Used

Table 1: Experimental Data for Ultramarine Red

<table>
<thead>
<tr>
<th>Sample</th>
<th>Absorption Wavelengths (nm)</th>
<th>Transmission Wavelengths (nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>400, 600, 800, 1000</td>
<td>200, 300, 400, 500</td>
</tr>
<tr>
<td>2</td>
<td>300, 500, 700, 900</td>
<td>100, 200, 300, 400</td>
</tr>
</tbody>
</table>

Electronic Spectra

The electronic spectra were computed for the structures that were obtained from the ultramarine red chromophore. The calculations were performed using the Gaussian 09 software, which is a commercial package for computational quantum chemistry.

Vibrational Results of the UHF SCF-311G** MP2 Calculations

The vibrational frequencies were calculated using the Gaussian 09 software. The calculations were performed on the structures obtained from the ultramarine red chromophore.

Conclusions and Possible Mechanism for Ultramarine Red Chromophore

In summary, the ultramarine red chromophore is a complex system that involves the interaction of several molecules. The electronic spectra and vibrational frequencies provide important information about the structure of the chromophore. Further experiments are needed to understand the mechanism of the ultramarine red chromophore.

Appendix

A19
Vanadium in the Mullite Structure

Andreas A Landman, Danita de Waal

Department of Chemistry, University of Pretoria, Pretoria, 0022,alandman@postino.up.ac.za, dawald@postino.up.ac.za

Abstract

Vanadium is a divalent transition metal found in mullite structures. The structural role of vanadium in mullite was investigated using scanning electron microscopy (SEM), X-ray diffraction (XRD), Raman spectroscopy, and electron probe microanalysis (EPMA). The results showed that vanadium substitutes for aluminium in the mullite structure, leading to the formation of vanadium mullite. The effects of vanadium on the properties of mullite were also discussed.

Keywords: mullite, vanadium, scanning electron microscopy, X-ray diffraction, Raman spectroscopy, electron probe microanalysis.

References


Appendix A20

Using Raman spectroscopy to verify pigment identity

D de Waal¹, AA Landman², RA Kruger² and J van Schalkwyk²

¹Department of Chemistry, University of Pretoria, Pretoria, 0002, Republic of South Africa;
²E-Mails: ddewaal@science.up.ac.za, landman@science.up.ac.za
Solawellle, PO Box 817, Randburg, 2125, Republic of South Africa

Introduction

Raman spectroscopy is a powerful tool for the analysis of solid-state materials due to its ability to provide detailed information about the vibrational and rotational modes of molecules. This technique is particularly useful in the identification of pigments and other materials used in the arts and crafts, as it can distinguish between different chemical compounds based on their unique vibrational fingerprints.

In this study, Raman spectroscopy was employed to verify the identity of ultramarine pigments. Ultramarine is a valuable pigment used in various applications, including fine art, and its authenticity is crucial to ensure the quality and longevity of artworks.

Methods

The Raman spectra of the pigments were recorded using a Renishaw 1000 Raman microscope. The samples were excited with a 532 nm laser, and the spectra were collected over a range of 100-3500 cm⁻¹. The spectral resolution was set to 1 cm⁻¹, and the integration time was 10 seconds.

Results

The Raman spectra of the pigments exhibited characteristic bands that corresponded to the vibrational modes of the pigment's chemical structure. These bands were used to identify the pigment and verify its authenticity.

Discussion

The Raman spectroscopy results confirmed the identity of the ultramarine pigments, providing evidence that they were of high quality and authenticity. This method offers a non-invasive and non-destructive approach to pigment identification, which is particularly valuable in the preservation of artworks.

Conclusions

Raman spectroscopy is an effective tool for the identification and authentication of ultramarine pigments. Its non-invasive nature makes it ideal for the analysis of artworks, ensuring the preservation of both historical and aesthetic value.

Acknowledgments

The authors would like to thank [Names] for their contributions to this research.

References


Table 1: Raman bands (cm⁻¹) of ultramarine pigments

<table>
<thead>
<tr>
<th>Pigment</th>
<th>1600 cm⁻¹</th>
<th>1800 cm⁻¹</th>
<th>2000 cm⁻¹</th>
<th>2400 cm⁻¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultramarine</td>
<td>1600 cm⁻¹</td>
<td>1800 cm⁻¹</td>
<td>2000 cm⁻¹</td>
<td>2400 cm⁻¹</td>
</tr>
</tbody>
</table>

Table 2: Intensity ratio of S₂ to S₃

<table>
<thead>
<tr>
<th>Pigment</th>
<th>S₂/S₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultramarine</td>
<td>S₂/S₃</td>
</tr>
</tbody>
</table>

Figure 1: Raman spectra of pigments

Figure 2: Intensity ratio of S₂ to S₃

Figure 3: Spectral intensity ratios

Appendix A23

Appendix

A24
E. BENEFICIARIES

E1. SPHERE-FILL (PTY) LTD

"Sphere-Fill (Pty) Limited is a wholly owned subsidiary of Ash Resources, South Africa. Ash Resources currently sells over 1,2 million tons/annum of the finest quality fly ash in Southern Africa and the Middle East." (Contact Details - Company profile, Sphere-Fill, http://www.superpozz.com/contact1.html, 2002, downloaded 10 September 2002 - 10h00.) The spherical form of fly ash makes it potentially functional in improving the workability of plastics. The current grey colour of fly ash is however a deterrent, since this colour hides the colour of added pigments and fly ash is not marketable in its grey state. Sphere-Fill (Pty) Ltd needed to know whether it is possible to colour fly ash. The new colours observed in fly ash will make fly ash more marketable, especially the heat treated fly ash. Any company utilising the proposed pigment syntheses will be expected to sign a contract with Sphere-Fill (Pty) Ltd as exclusive supplier of fly ash.

E2. ROLFES COLOUR PIGMENTS INTERNATIONAL

Rolfes Colour Pigments International is interested in producing ultramarine blue locally. They can now consider using fly ash to produce the pigment.

E3. UNIVERSITY OF PRETORIA

"The University of Pretoria's mission is to:

- be an internationally recognised academic institution which provides teaching, undertakes research and renders community service;
- fulfil the educational, cultural, social, economic and technological needs of the South African and Southern African communities; and

This project aids in achieving these goals by:

- Research that is relevant to local companies
- Research output, with international exposure, in the form of journal articles (1 published, 1 accepted, 2 submitted and 4 prepared), conference contributions in the form of posters (9) and lectures (4) and a thesis/dissertation.
- Research grants help in extending and improving the facilities at the University of Pretoria.

E4. SCIENCE

Fly ash is a powerful starting reagent in the synthesis of ultramarine blue, and could possibly be used in other applications as well. Science gained from this research a methodology for studying the solid-state reactions of fly ash.

E5. ENVIRONMENT

Man harnesses energy for beneficial purposes. Unfortunately this often has a price. In the Republic of South Africa the price of using low grade, abundantly available coal in generating electricity is mountains of fly ash. The aim of this project is to find other uses for fly ash and therefore reduce the problem of unused resources that pose environmental problems. Less fly ash in the environment will benefit not only the Republic of South Africa but also all countries that use coal-fired power plants.
Aspects of solid-state chemistry of fly ash and ultramarine pigments