In-Source Decay during Matrix-Assisted Laser Desorption/Ionization Combined with the Collisional Process in an FTICR Mass Spectrometer

Daiki Asakawa, David Calligaris, Tyler A. Zimmerman, and Edwin De Pauw*

Mass Spectrometry Laboratory, Department of Chemistry, and GIGA-Research University of Liège, B-4000 Liège (Sart-Tilman), Belgium

SUPPORTING INFORMATION

The batch processing script for exporting peak lists is found on page 2, the source code for MatrixPeaksRemover.java (to remove matrix peaks) is on pages 3-8, and the source code for MatrixSubtractImaging.java (to plot ion images) is on pages 9-12.

The batch processing script for use with DataAnalysis software (Bruker Daltonics) is a simple four line script that results in the peak lists for all of the spectra from a FTMS imaging dataset to be exported into text files. Each of the text files is labeled with the X-Y coordinates of the spectrum location on the tissue section. The number 5 in the third line of code means that the sixth portion (determined by number of back slashes) of the complete path of a spectrum folder will be used as the file name of each exported peak list.

```
Sub Form_OnLoad

   Analysis.Spectra(1).MassListFind 100, 2600

   Analysis.Spectra(1).ExportMassList "D:\users\Tyler\" + Split(Analysis.Path, "\")(5) + ".txt", daASCII

End Sub
```

The MatrixPeaksRemover code is used to remove MALDI matrix peaks from the peak lists of an MS imaging dataset, thereby allowing enhanced spectral interpretation without the interference of matrix cluster ion peaks.

```
/*
 * MatrixPeaksRemover.java
 *
 * Created on 22-feb.-2012, 16:49:26
 */

/**
 *
 * @author Tyler A. Zimmerman, Ph. D.
 */

import java.io.*;
import javax.imageio.*;
import java.awt.image.*;
import java.awt.*;
import java.lang.*;
import java.lang.Math.*;
import java.util.*;
import java.util.Arrays.*;
import java.util.Collections;
import java.io.File;

public class MatrixPeaksRemover
{
        public static void main(String args[ ]) throws IOException
                {

                String[] filename = new String[3];
                filename[0]="D:\\Tyler";                        //Path of the working directory (where all the files are)
                filename[1]="Matrix peak list for 1-5 DAN.txt"; //Text file containing the peak list of the matrix alone
                filename[2]="Processed spectra text files";     //The name of the peak list containing directory (within the working directory)
                ReadLines(filename);
                }

static void ReadLines(String[] filename) throws IOException
                {

                String WorkingPath = filename[0];

                 if (WorkingPath.endsWith("\\") == false | WorkingPath.endsWith("/") == false)
                {
                WorkingPath = WorkingPath + "\\";                                   //Ensures a slash in the path
                }                                                                   //
                                                                                    //
                String userDir = WorkingPath + filename[2];                         //Creates complete path
                                                                                    //to the peak list directory
                File dir = new File(userDir);                                       //
                                                                                    //
                String[] childrenTemp = dir.list();                                 //Gathers file names
                                                                                    //of text files
                String[] children = new String[childrenTemp.length];                //
                                                                                    //
                for (int i = 0; i <= childrenTemp.length-1; i++)                    //Creates complete path
                {                                                                   //to peak lists
```

```
55                  children[i] = WorkingPath + filename[2] + "\\" + childrenTemp[i];      //
56          }                                                                               //
57                                                                                          //
58          String[] SplitName = new String[5];                                             //This next step splits
59                                                                                          //the file names so as
60          double[][] Coord = new double[children.length][2];                              //to recover the X-Y
61                                                                                          //coordiniates of the
62          for (int i = 0; i <= children.length-1; i ++)                                   //peak lists from
63          {                                                                               //their file names
64                  SplitName = children[i].split("0X");                                    //
65                                                                                          //The structure of this whole section of code
66                  if (SplitName.length > 1)                                                       //will reflect the actual file structure
67                  {                                                                               //
68                  Coord[i][0] = Integer.valueOf(SplitName[1].substring(0, 3)).intValue();         //The X-Y positions
69                  Coord[i][1] = Integer.valueOf(SplitName[1].substring(4, 7)).intValue();         //are assigned to the
70                  }                                                                               //variable "Coord"
71          }                                                                                       //
72
73          int Rownum = 0;
74          int a = 0;
75                                                                                                  //
76          Double MassList[][] = new Double[children.length][5000];                            //The variable MassList has
77          Double Intensity[][] = new Double[children.length][5000];                           //dimensions of the total
78                                                                                                  //number of peak lists, by
79          for (int i = 0; i <= children.length-1; i++)                                        //a number large enough
80          {                                                                                   //to include all the
81                  Rownum = 0;                                                                 //individual peaks.
82                                                                                                          //Same dimensions for the
83                  LineNumberReader lnrM = new LineNumberReader(new FileReader(children[i]));               //variable Intensity.
84                  lnrM.setLineNumber(1);                                                                  //All values are read
85                  StreamTokenizer stokM = new StreamTokenizer(lnrM);                                      //from the peak lists
86                  stokM.parseNumbers();                                                                   //conatining text files.
87                  stokM.eolIsSignificant(true);                                                           //
88                  stokM.nextToken();
89
90                  StreamTokenizer stok = new StreamTokenizer(lnrM);
91
92                          while (stokM.ttype != StreamTokenizer.TT_EOF)
93                          {
94                                  while (stokM.ttype != StreamTokenizer.TT_EOF)
95                                  {
96                                          for (int colnum=1; colnum<=4; colnum++)
97                                                  {
98                                                          if (Rownum >= 1)
99                                                          {
100                                                         if (colnum==1) {MassList[i][Rownum-1]=stokM.nval;}
101                                                         if (colnum==2) {a = 0;}
102                                                         if (colnum==3) {Intensity[i][Rownum-1]=stokM.nval;}
103                                                         }
104
105                                                         if (colnum==4) {stokM.nextToken();
106                                                                         Rownum++;
107                                                                         break;}
108                                                         stokM.nextToken();
109                                                         }
110                                 if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
111                                 }
112                         if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
113                         }
```

```
114                 }
115                                                                                 //
116                 Double MassListRevSort[][] = new Double[MassList.length][5000];  //The values are sorted in descending order
117                 Double IntensityRevSort[][] = new Double[MassList.length][5000]; //to allow for more accuate ISD sequencing
118                                                                                 //from the cleaned peak lists that will
119                 int LengthCount = 0;                                            //eventually be outputted from this code.
120                                                                                 //See Zimmerman et al., Anal. Chem. 2011, 83, 6090.
121                 for (int i = 0; i <= MassList.length-1; i++)                    //
122                 {
123                         for (int j = 0; j <= 5000-1; j ++)
124                         {
125                                 if (MassList[i][j] == null)
126                                 {
127                                         LengthCount = j-1+1;
128                                         break;
129                                 }
130                         }
131
132                         if (LengthCount > 5)
133                         {                                                                       //
134                                 Double MassListItemp[] = new Double[LengthCount];       //This creates temporary MassList and Intensity
135                                 Double IntensityItemp[] = new Double[LengthCount];      //variables that are first sorted and then
136                                                                                         //entered into new arrays called MassListRevSort
137                                 for (int j = 0; j <= MassListItemp.length-1; j++)       //and IntensityRevSort.
138                                 {                                                       //
139                                         MassListItemp[j] = MassList[i][j];
140                                         IntensityItemp[j] = Intensity[i][j];
141                                 }
142
143                         for (int lastPlace = MassListItemp.length-1; lastPlace > 0; lastPlace--)
144                                 {
145                                 int minLoc = 0;                                 //
146                                                                                 //The actual sorting takes place inside this for loop.
147                                         for (int k = 0; k <= lastPlace; k++)    //
148                                         {
149                                         if (MassListItemp[k] < MassListItemp[minLoc])
150                                         {
151                                                 minLoc = k;
152                                         }
153                                         }
154
155                                 Double Temp = MassListItemp[minLoc];
156                                 MassListItemp[minLoc] = MassListItemp[lastPlace];
157                                 MassListItemp[lastPlace] = Temp;
158
159                                 Double Temp1 = IntensityItemp[minLoc];
160                                 IntensityItemp[minLoc] = IntensityItemp[lastPlace];
161                                 IntensityItemp[lastPlace] = Temp1;
162                                 }
163
164                                 for (int j = 0; j <= MassListItemp.length-1; j++)
165                                 {
166                                         MassListRevSort[i][j] = MassListItemp[j];
167                                         IntensityRevSort[i][j] = IntensityItemp[j];
168                                 }
169                         }
170                 }
171                                                                                                         //
172                 BufferedReader inM = new BufferedReader(new                                             //The same steps to read the MS
```

```
173                 FileReader(WorkingPath + filename[1]));                                     //imaging peak lists are applied here to
174                 int MlineCount = 0;                                                         //read the single text file containing
175                                                                                             //the peak list of the MALDI matrix
176                 do                                                                          //alone.
177                 {                                                                           //
178                 String lineM = inM.readLine();                                              //
179                         if (lineM == null) break;
180                 MlineCount++;
181                 } while (true);
182
183                 Double MatrixPeaks[] = new Double[MlineCount];
184
185                 int rownum = 0;
186
187                 LineNumberReader lnrM = new LineNumberReader(new FileReader(WorkingPath + filename[1]));
188                 lnrM.setLineNumber(1);
189                 StreamTokenizer stokM = new StreamTokenizer(lnrM);
190                 stokM.parseNumbers();
191                 stokM.eolIsSignificant(true);
192                 stokM.nextToken();
193
194                 StreamTokenizer stok = new StreamTokenizer(lnrM);
195
196                         while (stokM.ttype != StreamTokenizer.TT_EOF)
197                         {
198                                 while (stokM.ttype != StreamTokenizer.TT_EOF)
199                                 {
200                                         for (int colnum=1; colnum<=2; colnum++)
201                                                 {
202                                                         if (colnum==1) {MatrixPeaks[rownum]=stokM.nval;}
203
204                                                         if (colnum==2) {stokM.nextToken();
205                                                                         rownum++;
206                                                                         break;}
207                                                         stokM.nextToken();
208                                                 }
209
210                                 if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
211                                 }
212
213                         if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
214                         }
215
216                 int RemovedCount = 0;                                                       //
217                                                                                             //This for loop is where the actual
218                 for (int i = 0; i <= MassListRevSort.length-1; i++)                         //matrix peak subtraction takes place
219                 {                                                                           //from the values in MassListRevSort,
220                         for (int j = 0; j <= 5000-1; j++)                                   //by setting the values equal to zero
221                         {                                                                   //if there is a peak that corresponds to
222                                 for (int k = 0; k <= MatrixPeaks.length-1; k++)             // a matrix peak.
223                                 {
224                                         if (MassListRevSort[i][j] != null && MassListRevSort[i][j] <= MatrixPeaks[k] + 0.001 &&
225                                         MassListRevSort[i][j] >= MatrixPeaks[k] - 0.001)
226                                         {
227                                                 MassListRevSort[i][j] = 0.0;                //
228                                                 IntensityRevSort[i][j] = 0.0;               //The values above of +/-0.001 Da
229                                                 RemovedCount++;                             //define the mass window around each
230                                         }                                                   //MALDI matrix peak that is used to
231                                 }                                                           //perform subtraction on all the MS
```

```
232                 }                                                   //imaging peak lists. This value can
233         }                                                           //be edited by the user.
234                                                     //
235         Double BinsIntensity[] = new Double[40000];     //Now begins the code for creating the TIC (total ion count)
236         int BinsIntensityCount[] = new int[40000];      //spectrum that is created by summing all the MS imaging peak
237                                                         //list intensities into appropriate sized mass bins. If creating a TIC using
238         for (int i = 0; i <= BinsIntensity.length-1; i++)   //0.01 Da bins over a 40 Da range, the number of bins needed is 40,000.
239         {                                               //
240                 BinsIntensity[i] = 0.0;                 //The BinsIntensity variable is initialized with zeros in place of
241         }                                               //of null values that tend to cause errors, in this for loop.
242                                                         //
243         Double MassIncrement = 600.0;                   //The MassIncrement variable is designed to increase in increments
244         int MassIncrementCount = 0;                     //over the mass range desired, starting with this value.
245                                                         //These for loops do the actual summing of peak intensities over the entire MS
246         for (int i = 0; i <= BinsIntensity.length-1; i++)   // imaging dataset based on the parameters selected
247         {                                               // based on the example case of using a 0.01 Da bin size.
248                 for (int j = 0; j <= MassListRevSort.length-1; j++)
249                 {
250                         for (int k = 0; k <= 5000-1; k++)
251                         {
252                                 if (MassListRevSort[j][k] != null && MassListRevSort[j][k] != 0.0 && IntensityRevSort[j][k] != null
253                                 && IntensityRevSort[j][k] != 0.0 && MassListRevSort[j][k] <= MassIncrement + 0.01 &&
254                                 MassListRevSort[j][k] >= MassIncrement)
255                                 {
256                                         BinsIntensity[MassIncrementCount] += IntensityRevSort[j][k];
257                                         BinsIntensityCount[MassIncrementCount]++;
258                                 }
259                         }
260                 }
261                                                                 //
262                 MassIncrement += 0.01;                          //This part of the code increases the MassIncrement over the entire
263                 MassIncrementCount++;                           // mass range of 40 Da. The MassIncrementCount increases the
264                                                                 // array indicies over sucessive runs of the for loop.
265                 System.out.println(MassIncrementCount);         //This code creates a visual printout in the MS-DOS
266                 System.out.println(MassIncrement);              //command prompt of the progress estimated.
267                 System.out.println("--------");                 //through the for loop, so that the computational time can be
268         }                                                       //estimated.
269
270         FileOutputStream out1;                                              //
271         PrintStream p1;                                                     //This code creates a new text file in the working
272         out1 = new FileOutputStream(WorkingPath + "TICMassList" + ".txt");  //directory and
273         p1 = new PrintStream( out1 );                                       //prints the TIC spectrum to this text file called
274                                                                             //TICMassList.txt, the value of which can be
275         for (int i = 0; i <= BinsIntensity.length-1; i++)                   // plotted in M/S Excel
276         {                                                                   //for visialization of the TIC spectrum.
277                 if (BinsIntensity[i] != null)
278                 {
279                         p1.println(BinsIntensity[i]);
280                 }
281         }
282         p1.close();
283                                                                 //
284         for (int i = 0; i <= MassListRevSort.length-1; i++)     //This code creates a text file in the working directory for each X-Y
285         {                                                       // position of the tissue MS imaging dataset, and prints
286                 FileOutputStream out;                           // the corresponding "cleaned" peak lists to each text file,
287                 PrintStream p;                                  //where MALDI matrix peak have been removed. Then, the code
288                                                                 //ends
289                 out = new FileOutputStream(WorkingPath + "X" + Integer.toString((int)Coord[i][0]) + "_Y" +
290                 Integer.toString((int)Coord[i][1]) + ".txt");
```

```
291                                 p = new PrintStream( out );
292
293                                 for (int j = 0; j <= 5000-1; j++)
294                                 {
295                                         if (MassListRevSort[i][j] != null)
296                                         {
297                                         if (MassListRevSort[i][j] != 0.0)
298                                         {
299                                                 p.println(MassListRevSort[i][j] + "\t" + IntensityRevSort[i][j]);
300                                         }
301                                         }
302                                 }
303
304                                 p.close();
305                         }
306                 }
```

The MatrixSubtractImaging software is used to plot select 2D ion images from an MS imaging dataset using the peak lists, and within a user-defined mass window. Plotting of ion images with this software can be done either before or after use of the above software to remove MALDI matrix peaks.

```
/*
 * MatrixSubtractImaging.java
 *
 * Created on 22-feb.-2012, 11:41:14
 */

/**
 *
 * @author Tyler A. Zimmerman, Ph. D.
 */

import java.io.*;
import javax.imageio.*;
import java.awt.image.*;
import java.awt.*;
import java.lang.*;
import java.lang.Math.*;
import java.util.*;
import java.util.Arrays.*;
import java.util.Collections;
import java.io.File;

public class MatrixSubtractImaging
{
        public static void main(String args[ ]) throws IOException
                {

                String[] filename = new String[5];
                filename[0]="D:\\Tyler";            //Path of the working directory (where all the files are)
                filename[1]="Example.bmp";          //The user chooses an output image name for the ion images
                filename[2]="Text files";           //The name of the peak list containing directory (within the working directory)
                filename[3]="630.32";               //The theoretical mass of the analyte to plot an ion image of
                filename[4]="0.2";                  //The mass tolerance window for plotting ion images
                ReadLines(filename);
                }

static void ReadLines(String[] filename) throws IOException
                {

                if (!(filename[1].toLowerCase().endsWith(".bmp")) && !(filename[1].toLowerCase().endsWith(".png"))  //
                        && !(filename[1].toLowerCase().endsWith(".jpg")))                                          //Tests image format
                        {System.out.println("Error: ouput image can only end in .jpg, .png, or .bmp");}            //

                String WorkingPath = filename[0];

                if (WorkingPath.endsWith("\\") == false | WorkingPath.endsWith("/") == false)
                {                                                                                  //
                WorkingPath = WorkingPath + "\\";                                                  // Ensures a slash in the path
                }                                                                                  //
                                                                                                   //
                String userDir = WorkingPath + filename[2];                                        //Creates complete path
                                                                                                   //to the peak list directory
                File dir = new File(userDir);                                                      //
                                                                                                   //
                String[] childrenTemp = dir.list();                                                //Gathers file names
                                                                                                   //of text files
                String[] children = new String[childrenTemp.length];                               //
```

```
                                                                                    //
for (int i = 0; i <= childrenTemp.length-1; i++)                                    //Creates complete path
{                                                                                   //to peak lists
        children[i] = WorkingPath + filename[2] + "\\" + childrenTemp[i];           //
}                                                                                           //
                                                                                            //This next step splits
String[] SplitNameX = new String[2];                                                        //the file names so as
String[] SplitNameX1 = new String[2];                                                       //to recover the X-Y
String[] SplitNameY = new String[2];                                                        //coordiniates of the
String[] SplitNameY1 = new String[2];                                                       //peak lists from
                                                                                            //their file names
double[][] Coord = new double[children.length][2];                                          //

for (int i = 0; i <= children.length-1; i ++)
{
        SplitNameX = children[i].split("X");
        SplitNameY = children[i].split("_Y");

        System.out.println(Arrays.toString(SplitNameX));
        System.out.println(Arrays.toString(SplitNameY));

        if (SplitNameX.length > 1 && SplitNameY.length > 1)
        {
        if (SplitNameX[1].split("_").length > 1)
        {

        SplitNameX1 = SplitNameX[1].split("_");
        SplitNameY1 = SplitNameY[1].split("\\.");
                                                                                    //
        Coord[i][0] = Integer.valueOf(SplitNameX1[0]).intValue();                   //The X-Y positions
        Coord[i][1] = Integer.valueOf(SplitNameY1[0]).intValue();                   //are assigned to the
        }                                                                           //variable "Coord"
        }                                                                           //
}

int Rownum = 0;
int a = 0;
                                                                                            //
Double MassList[][] = new Double[children.length][5000];                                    //The variable MassList has
Double Intensity[][] = new Double[children.length][5000];                                   //dimensions of the total
                                                                                            //number of peak lists by
for (int i = 0; i <= children.length-1; i++)                                                //a number large enough
{                                                                                           //to include all the
        Rownum = 0;                                                                         //individual peaks.
                                                                                            //Same dimensions for the
        LineNumberReader lnrM = new LineNumberReader(new FileReader(children[i]));           //variable Intensity.
        lnrM.setLineNumber(1);                                                              //All values are read
        StreamTokenizer stokM = new StreamTokenizer(lnrM);                                  //from the peak lists
        stokM.parseNumbers();                                                               //conatining text files.
        stokM.eolIsSignificant(true);                                                       //
        stokM.nextToken();

        StreamTokenizer stok = new StreamTokenizer(lnrM);

                while (stokM.ttype != StreamTokenizer.TT_EOF)
                {
                        while (stokM.ttype != StreamTokenizer.TT_EOF)
                        {
                                for (int colnum=1; colnum<=3; colnum++)
                                        {
                                                if (colnum==1) {MassList[i][Rownum]=stokM.nval;}

```

```
120                                                                     if (colnum==2) {Intensity[i][Rownum]=stokM.nval;}
121
122                                                                     if (colnum==3) {stokM.nextToken();
123                                                                                     Rownum++;
124                                                                                     break;}
125
126                                                                     stokM.nextToken();
127                                                             }
128                                     if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
129                                     }
130                             if (stokM.ttype == StreamTokenizer.TT_EOF) {break;}
131                             }
132                     }
133
134                     Double MassCenter = Double.valueOf(filename[3]);                        //
135                     Double MassWindow = Double.valueOf(filename[4]);                        //The appropriate mass window
136                     Double MassLow = MassCenter - MassWindow;                               //for ion image plotting
137                     Double MassHigh = MassCenter + MassWindow;                              //is defined with these
138                                                                                             //variables.
139                     double MaxX = 0;                                                        //
140                     double MaxY = 0;
141
142                     for (int i = 0; i <= Coord.length-1; i++)                       //
143                     {                                                               //The maximum X and Y
144                             if (Coord[i][0] > MaxX)                                 //coordinates are found
145                             {                                                       //in order to output an
146                                     MaxX = Coord[i][0];                             //image of appropriate size.
147                             }                                                       //
148
149                             if (Coord[i][1] > MaxY)
150                             {
151                                     MaxY = Coord[i][1];
152                             }
153                     }
154
155                     int width = (int)MaxX;
156                     int height = (int)MaxY;
157
158                     String format = "JPG";                                          //
159                                                                                     //The type of image to output
160                     if (filename[1].toLowerCase().endsWith(".png"))                 //is taken from the user input
161                                             {format = "PNG";}                       //at the very top of this code.
162                                                                                     //
163                     if (filename[1].toLowerCase().endsWith(".bmp"))
164                                             {format = "BMP";}
165
166                     BufferedImage image = new BufferedImage(width + 10, height + 10, BufferedImage.TYPE_INT_RGB);
167
168                     Double IntensityMap[] = new Double[Coord.length];
169
170                     for (int i = 0; i <= MassList.length-1; i++)                    //
171                     {                                                               // This tests if the mass resides within the user-
172                             for (int j = 0; j <= 5000-1; j++)                       //defined region of interest.
173                             {                                                       //
174                                     if (MassList[i][j]!= null && MassList[i][j] >= MassLow && MassList[i][j] <= MassHigh)
175                                     {
176                                             IntensityMap[i]= Intensity[i][j];
177                                     }
178                             }
179                     }
180
181                     double MaxIntensity = 0;
```

```
182                 double IntensitySum = 0;
183
184                 double MaxInd = 0;
185
186                 for (int i = 0; i <= IntensityMap.length-1; i++)
187                 {
188                         if (IntensityMap[i] != null && IntensityMap[i] > MaxIntensity)
189                         {                                                       //The maximum intensity value
190                                 MaxIntensity = IntensityMap[i];                 //throughout the dataset in
191                                 MaxInd = i;                                     //the user-defined mass region
192                         }                                                       //is found.
193                 }                                                               //
194                                                                                 //
195                 String[] SplitTemp = filename[1].split("\\.");                  //Important values are added to the output image name
196
197                 String OutImageName = SplitTemp[0] + "_mz" + MassCenter + "+-" + MassWindow + "_Imax" + (int)MaxIntensity + "." +
198                 SplitTemp[1];
199
200                 double colorValue = 0;
201
202                 for (int i = 0; i <= IntensityMap.length-1; i++)
203                 {
204                 if (IntensityMap[i] != null)
205                 {                                                               //
206                         colorValue = IntensityMap[i] / (MaxIntensity);          //The color value is scaled to the
207                                                                                 //maximum intensity value found above.
208                         if (colorValue > 1)                                     //
209                         {                                                       //The color values are normalized to 1.
210                                 colorValue = 1;                                 //
211                         }                                                       //
212
213                         Graphics graphics = image.getGraphics();
214                         Color color = new Color((int)(colorValue * 255), (int)(colorValue * 255), (int)(colorValue * 255));
215                         graphics.setColor(color);
216                         graphics.fillRect((int)Coord[i][0] -1, (int)Coord[i][1] - 1, 1, 1);             //
217                 }                                                                                       //A 2D ion image
218                 }                                                                                       //is outputted
219                                                                                                         //and the code
220                 ImageIO.write(image, format, new File(WorkingPath + OutImageName));                     //ends.
221         }                                                                                               //
```